

PACE Handbuch

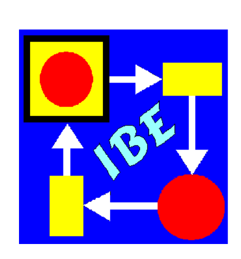
Modellierung und Simulation

PACE 2008

IBE

Simulation Engineering GmbH

2008



IBE Simulation Engineering GmbH
Postfach 1142
D-85623 Glonn
Tel.: +49-8093-5000
Fax: +49-8093-902687
E-mail: info@ibepace.com
Home: http://www.ibepace.com

Dieses Handbuch ist gültig für PACE 2008.

Inhaltsverzeichnis

1 EINLEITUNG

1.1 Das Software-Tool PACE

„If you can't describe what you are doing as a process, you don't know what you are doing." Dieses Zitat von W. Edwards Deming, der schon Mitte des letzten Jahrhunderts als erster die Bedeutung der Qualitätssicherung erkannt hat, skizziert eine wichtige Voraussetzung für den erfolgreichen Aufbau und Betrieb von Prozessen aller Art, wie Geschäftsprozessen, Produktionsprozessen, Logistikprozessen, usw. Um die statische Struktur und die Dynamik eines Prozesssystems darzustellen und zu optimieren, ist es zweckmäßig ein möglichst realistisches Modell des Systems zu erstellen, das sich auf einem Rechner ausführen lässt. Ein ausführbares (animierbares) und realitätsnahes Modell ermöglicht kostengünstige Experimente, mit denen zum Beispiel der Systementwurf verifiziert oder die optimale Parametrierung hinsichtlich bestimmter Kriterien (Produktionskosten, usw.) gefunden werden kann.

PACE ist ein graphisches Software-Tool für die Entwicklung ausführbarer Modelle von ereignis-orientierten, parallelen, diskreten Systemen. Modelle werden in PACE mit der halbgrafischen Modellierungsprache MSL (Modeling and Simulation Language) entwickelt, die auf attributierten hierarchischen Petri-Netzen basiert. Das dynamische Verhalten eines Modells bzw. Systems kann mit dem PACE-Simulator animiert, visualisiert, analysiert und gegebenenfalls optimiert werden. In technischen Anwendungen kann PACE auch auf verschiedene Weisen zur Steuerung von Anwendungen eingesetzt werden.

Hierarchische attributierte Petri-Netze werden verwendet, um sowohl die Struktur eines Modells als auch die Abfolge der parallel ablaufenden Prozesse zu modellieren. Mit ihnen lassen sich Zustände und

Zustandsübergänge transparent darstellen. Sie eignen sich deshalb ganz besonders gut, um ereignisorientierte, parallele Systeme wie Prozeßsteuerungs-Systeme, Kommunikationsprotokolle oder Geschäftsprozesse zu beschreiben.

Die Daten eines Systems und deren Verarbeitung werden auf der anderen Seite durch Smalltalk-Objekte und Smalltalk-Code-Sequenzen modelliert. Daten können an durch das Netz fließende Marken angeheftet und dadurch an die Orte im Netz transportiert werden, an denen sie für Berechnungen benötigt werden. Für die Simulation und die direkte Programmausführung mit PACE stehen grundsätzlich alle Klassen und Botschaften einer umfangreichen Smalltalk-Bibliothek zur Verfügung. Für die allermeisten System-Modelle wird jedoch nur eine verschwindend geringe Anzahl davon benötigt.

Die graphischen Darstellungsmittel, mit denen in PACE System-Modelle aufgebaut werden, bestehen aus einer kleinen Menge standardisierter Komponenten, die vom Entwickler durch Anhängen von Smalltalk-Inskriptionen an die jeweilige Aufgabe angepasst werden. Der dynamische Objektfluß innerhalb eines Modells wird damit von der Bedeutung der verwendeten Objekte in der realen Welt losgelöst. Mit ihnen lassen sich deshalb System-Modelle für unterschiedlichste Anwendungen entwerfen.

Welche Bedeutung die einzelnen Sprach-Elemente von PACE im jeweiligen Anwendungsgebiet haben, legt der Anwender von Fall zu Fall selbst fest. So können beispielsweise die dynamischen Objekte, die sich innerhalb eines PACE-Netzes bewegen (sog. Marken) im Modell eines Kommunikations-Systems eine Botschaft bedeuten, während sie im Modell eines Produktions-Systems irgendein Produkt oder Material auf einem Förderband darstellen.

Um dieses Einsatz-Spektrum abzudecken, enthält die PACE-Systembeschreibungs-Sprache MSL allgemeine graphische und textuelle Fähigkeiten,

- um den internen Zustand von Objekten zu beschreiben,
- um festzuhalten, unter welchen Bedingungen sich das Objekt von einem Ort zum anderen bewegt,

- um festzustellen, wieviel Zeit ein solcher Verschiebungsvorgang benötigt,
- um eine Auswahlstrategie zu bestimmen, wenn unterschiedliche Wege zur Verfügung stehen,
- um herauszufinden, auf welche Weise sich der interne Zustand eines Objekts verändert hat,
- um System-Spezifikationen zu modularisieren,
- um die Standard-Netzelemente durch anschauliche Bilder, die zu einem intuitiven Verständnis des Modellinhalts führen, zu ersetzen,

usw. Gerade der zuletzt genannte Punkt ist häufig von sehr großer Bedeutung, weil bei der Anwendung oder Begutachtung erstellter Modelle normalerweise nicht die letzten Implementierungsdetails zählen, sondern die Funktionalität des Models und die für die Bedienung bereitgestellte Nutzer-Oberfläche.

1.2 Warum modellieren mit Petri-Netzen?

Blickt man auf die Vorstellungen, die den Vorgehensweisen bei der Modellierung in den letzten 20 Jahren zugrunde liegen, so stellt man den Trend von prozeßorientierten zu objektorientierten Beschreibungformen fest. Die neuesten Verfahren, wie z.B. UML, verzichten ganz auf die prozeßorientierte Sicht und zerlegen alle zu modellierenden Sachverhalte in eine Menge von Objekten, die teilweise über verkettete Listen korreliert sind.

Nun ist es im Prinzip gleichgültig, welche Methode man bei der Modellierung einsetzt, wenn nur das zu modellierende System vollständig und adäquat dargestellt werden kann. Darunter wird folgendes verstanden:

- Ein System ist vollständig beschrieben, wenn alle für die jeweilige Aufgabe erforderlichen statischen und dynamischen Aspekte des Systems dargestellt sind.
- Ein System ist adäquat beschrieben, wenn die Struktur und Wirkungsweise des realen System und die des Modells, seinem Abbild, so weitgehend übereinstimmen, daß der Anwender, der normalerweise nicht an der Modellentwicklung teilgenommen hat, das Modell aufgrund der Systemkenntnis des realen Systems verstehen kann.

Die meisten Spezifikationssysteme ermöglichen die vollständige Beschreibung von Systemen, wobei die dynamischen Aspekte, besonders in den sog. CASE-Tools, häufig nicht formal dargestellt werden. Deshalb überprüfen die meisten Spezifikationssysteme nur statische Sachverhalte. So wird beispielsweise immer überprüft, ob Objekte, die referenziert werden, irgendwo deklariert sind. Die wenigsten Systeme bieten aber Möglichkeiten, um die Dynamik eines komplexen Modells zu überprüfen, häufig deshalb, weil die Art der Beschreibung eine solche Überprüfung schwierig oder unmöglich macht.

Das zweite Kriterium ist ein ingenieurmäßiger Gesichtspunkt. Damit ein Modell in der praktischen Arbeit verwendet werden kann, beispielsweise um das dynamische Verhalten des realen Systems zu verstehen und zu verifizieren, muß ein möglichst einfacher Zusammenhang zwischen Modell und Realität bestehen. Das erreicht man normalerweise, indem das Modell dem realen System sowohl statisch als auch dynamisch möglichst weitgehend nachgebildet wird.

Die rein objektorientierte Modellierung mag für viele Systeme die adäquate Beschreibungsmethode sein, sie entspricht aber oft nicht der natürlichen ingenieur-gerechten Sicht der Dinge. Eine adäquate Beschreibung sollte die Sachverhalte unbedingt so abstrahieren, daß die Charakteristiken des zu modellierenden Systems erhalten bleiben.

Bei der Betrachtung unserer Umwelt registriert man als grundlegende Charakteristiken die folgenden beiden Gegebenheiten:

- Objekte sind zeitunabhängige physikalische oder logische Größen, die im Modell ‘als Einheit’ behandelt werden können (Bäume, Autos, Briefe, Mitarbeiter, Formulare, usw.). Wie üblich können Objekte aus anderen Objekten zusammengesetzt werden.

- Prozesse sind zeitabhängige Vorgänge, die auf Objekte einwirken, d.h. Objekte transportieren und/oder in andere Objekte umwandeln. Wie üblich können Prozesse in Teilprozesse zerlegt werden.

Mit diesem Ansatz, der auch in anderen naturwissenschaftlichen Disziplinen verwendet wird,[1] ist eine Methodik charakterisiert, wie sie in den attributierten Petri-Netzen von PACE angewandt wird. So wird beispielsweise das statische Netzelement ‘Stelle’ zu Modellierung von Objekten herangezogen, die Speicherorte repräsentieren (Lager, Förderband, usw.). Transitionen können als verarbeitende

[1] Z.B.: Statik und Dynamik in der Mechanik, Elektrostatik und Elektrodynamik in der Elektrizitätslehre.

Objekte (Maschinen) angesehen werden. Das dynamische Objekt ‘Marke’ schließlich kann als ein Transportbehälter für zu bearbeitenden Objekte angesehen werden. Prozesse bzw. Teilprozesse werden durch das Schalten von Transitionen realisiert und im Animationsmodus durch fließende Marken im Netz visualisiert; ihre Richtung wird über Konnektoren festgelegt.

Wie die zahlreichen Beispiele, die der PACE-Auslieferung beigefügt sind, zeigen, kann man mit dieser einfachen Sichtweise reale Systeme ziemlich getreu auf virtuelle Systeme (Modelle) abbilden. Dabei ist es möglich, die Topologie des realen Systems weitgehend auf das Modell zu übertragen, wobei das Verständnis des Zusammenhangs zwischen Realität und Modell durch Einsetzen entsprechender Bilder statt der Standard-Grafiken für Stellen, Marken, usw. sehr stark gefördert wird. Der Aufbau des Systems wird durch Verfeinerung der Objekte dargestellt. Man modelliert das Systemverhalten an den Orten im Netz, an denen das Verhalten auftritt, durch Einfügen von Unternetzen, Einfüllen von Programmcode in Transitionen, usw.

1.3 Über dieses Manual

Dieses Manual steht sowohl in der vorliegenden index-orientierten Form, als auch in verkürzter Form als verfeinerndes Online-Manual zur Verfügung. Es wurde in viele kleine Kapitel zerlegt und so aufgebaut, daß es sowohl für das Erlernen von PACE, als auch als Nachschlagewerk verwendet werden kann.

Das Manual beschreibt die hinter PACE stehenden Ideen, alle PACE-Eigenschaften und die Implementation (soweit dies für den Anwender von Interesse sein kann). Eine weiteres Handbuch, die Smalltalk-Fibel, enthält eine kurze Einführung in die für PACE nötigen Sprachelemente von Smalltalk. Beispiele zu den Smalltalk-Sprachelementen findet man auch im vorliegenden Handbuch und im PACE-Kochbuch.

Es wird nicht empfohlen, sich durch das ganze Manual in einem Zug hindurch zu arbeiten. Die einzelnen Kapitel sind weitgehend voneinander unabhängig und können deshalb, wenn ihr Inhalt zunächst nicht gebraucht wird, übersprungen werden. Wichtig ist aber, daß man sich zunächst einen Überblick darüber verschafft, was in dem Manual grundsätzlich beschrieben ist und wo man bestimmte Sachverhalte, die man beim Erstellen eines Modells zu berücksichtigen hat, finden kann.

Eine erste grobe Übersicht kann man sich durch Blättern verschaffen. Es wurde versucht, die Kapitelüberschriften möglichst aussagekräftig zu gestalten.

Die meisten PACE-Features können einfach und intuitiv eingesetzt werden, wenn man erst einmal die Grundlagen der Bedienung von PACE kennt. Um sich die Verwendung der PACE-Sprachelemente einzuprägen und einzuüben, wird empfohlen, zunächst den PACE-Starter Schritt für Schritt nachzuvollziehen und während des erstmaligen Lesens die beschriebenen Sachverhalte direkt am Rechner auszuprobieren. Nur wenn man das Hilfsmittel gut beherrscht, kann

man sich später voll auf die eigentliche Arbeit, nämlich das Entwickeln von dynamischen Modellen, konzentieren.

Die PACE-Auslieferung enthält eine große Anzahl von Beispiel-Modellen, die teilweise aus Anwendungen von PACE stammen. Anhand dieser Modelle läßt sich studieren, wie bestimmte Problemstellung mit PACE modelliert werden.

2 DIE PACE-INSTALLATION

2.1 Inhalt des PACE-Verzeichnisses

Der PACE-Auslieferung liegt eine Kurzbeschreibung ‘liesmich.pdf’ bei, die unter anderem die bei der Installation auszuführenden Einzelschritte beschreibt. Die zu installierenden Dateien und Verzeichnisse liegen nach der Installation in einem Verzeichnis vor, das vom Anwender während der Installation gewählt wird und das im folgenden 'PACE-Verzeichnis'[2] genannt wird. Je nach Organisation des Zielsystems können die verschiedenen Dateien und Verzeichnisse nach der Installation an andere Orte in der Verzeichnisstruktur gebracht werden. Dabei müssen im PACE-Image (siehe unten) ggf. die Pfade zu den Verzeichnissen neu eingestellt werden.

Unmittelbar nach der Installation liegen im PACE-Verzeichnis folgende Dateien und Verzeichnisse vor:

2.1.1 Die virtuelle PACE-Maschine

Mit dem PACE-Hauptprogramm wird das im nächsten Abschnitt beschriebene PACE-Image ausgeführt. Das Name des PACE-Hauptprogramms lautet unter MS-Windows: 'pacevm.exe'.

Das PACE-Hauptprogramm ist eine sog. virtuelle Maschine und führt das im PACE-Image gespeicherte Modell aus. Das Hauptprogramm besteht aus mehreren Modulen, die dynamisch zu einem ausführbaren Programm gebunden werden. Unter Windows gibt es mindestens eine DLL (Dynamic Link Library) 'USERDLL.DLL'. Der

[2] Defaultmäßig wird das Verzeichnis ‘c:\programme\pace2008’ verwendet.

Anwender kann USERDLL.DLL erweitern und weitere DLLs erstellen. Er kann damit die Funktionalität der virtuellen Maschine nachträglich erweitern.

2.1.2 Das PACE-Image

Das Image beinhaltet den aktuellen Ausführungszustand von PACE und heißt nach der Installation 'pace2008.imm'. Der Anwender kann diesen Namen jedoch jederzeit ändern. Zu beachten ist allerdings, daß beim Abspeichern eines Image (Menüpunkte ‘store image’ oder ‘leave PACE’ im Menü 'file' der PACE-Hauptleiste) automatisch die Erweiterung '.imm' angehängt wird. Der Anwender braucht deshalb nur den Filenamen ohne Erweiterung einzugeben.

Der aktuelle Ausführungszustand ist der aktuelle Zustand von PACE einschließlich aller Daten, Fenster, u.s.w. Beim erneuten Starten von PACE kann genau an der Stelle weitergearbeitet werden, die beim vorherigen Abspeichern des Images vorlag.

2.1.3 Die Verzeichnisse 'nets' und ‘modules’

Außer dem Abspeichern als Image können Modelle ganz oder teilweise als Netze (Modelle) oder Unternetze (Module) in Files gespeichert werden. Die Files mit den Netzen und Unternetzen werden standardmäßig im Verzeichnis 'nets' oder ‘modules’ abgelegt. Diese Voreinstellungen können vom Anwender geändert werden. Sollte das jeweils angesprochene Verzeichnis nicht vorhanden sein, wird es von PACE automatisch als Unterverzeichnis des PACE-Verzeichnisses erzeugt.

Dateien mit Erweiterung '.net'
In Dateien mit der Erweiterung '.net' sind komplette Modelle gespeichert. Diese Dateien werden standardmäßig im Unterverzeichnis ‘nets’ des PACE-Verzeichnisses erzeugt, sobald das durch den 'store net'-Befehl im 'file'-Menü der PACE-Hauptleiste aufrufbare Auswahlmenü ausgeführt wird. In diesem Menü kann auch ein neuer Filename und ein anderes Verzeichnis für die Netzdatei vorgegeben werden.

Jede Auslieferung enthält zahlreiche Beispiel-Modelle, die teilweise aus früheren PACE-Anwendungen stammen.

Dateien mit Erweiterung '.sub'
In Dateien mit der Erweiterung '.sub' sind die Netz-Module gespeichert. Dabei handelt es sich um wiederverwendbare Teilnetze (Komponenten), die in verschiedenen Anwendungen einsetzbar sind. Diese Dateien werden für die im 'Net List'-Fenster markierten Module standardmäßig im Unterverzeichnis ‘modules’ erzeugt, sobald das durch den 'store module'-Befehl im 'file'-Menü der PACE-Hauptleiste aufrufbare Auswahlmenü ausgeführt wird. In diesem Menü kann auch ein neuer Filename und ein anderes Verzeichnis für die Moduldatei vorgegeben werden.

Das Modulverzeichnis enthält nach der Installation einige Standard-Module für spezielle Aufgaben. Beispielsweise gibt es einen Modul 'LIFO', den Marken in umgekehrter Ankunftsreihenfolge verlassen.

Für jedes Modell und jeden Modul, in dem individuelle (d.h. vom Anwender vorgegebene) Ikonen benutzt werden, wird eine Bild-Datei mit Erweiterung '.icn' erstellt, welche die Bitmaps der Ikonen speichert. Bild-Dateien werden standardmäßig im Netz-Verzeichnis bzw. dem Modul-Verzeichnis abgelegt.

2.1.4 Das Verzeichnis 'document'

Das Verzeichnis 'document' im PACE-Verzeichnis enthält die PACE-Handbücher und den PACE-Starter. Im Verzeichnis ‘documents’ auf der PACE-CD sind weitere Broschüren und Artikel über PACE gespeichert.

Der größte Teil der Dokumentation von PACE wird im PFD-Format ausgeliefert und kann z.B. mit dem Adobe-ACROBAT-Reader bequem am Bildschirm gelesen und nötigenfalls ausgedruckt werden.

In der Datei 'manual.con' ist das PACE-Online-Manual gespeichert. Es kann über das Help-Menü der PACE-Hauptleiste aufgerufen

werden und ermöglicht einen schnellen und bequemen Zugriff auf Informationen über PACE durch Verfeinerung.

2.1.5 Das Verzeichnis 'samples'

Hier sind in zugeordneten Unterverzeichnissen größere Modelle in geladener Form (als sog. Image) für die sofortige Ausführung gespeichert. Die geladenen Modelle sind in der 'Info zu den Samples'-Datei im Verzeichnis 'samples' kurz beschrieben. Falls Artikel über die Modelle vorliegen, wurden sie beigefügt..

2.1.6 Das Verzeichnis 'printing'

Hier werden die PostScript-Dateien abgelegt, welche durch Ausführen von Menü-Punkten im 'print'-Menü der PACE-Hauptleiste erstellt werden. Diese PostScript-Dateien werden mit dem Namen des Netzes benannt und mit der Erweiterung '.ps' versehen.

2.1.7 Das Verzeichnis 'ioutils'

In diesem Verzeichnis werden die Daten der Visualisierungsfenster mit der ‘store’-Menüanweisung des jeweiligen Fensters gespeichert. Werden diese Daten in ein Visualisierungsfenster des entsprechenden Typs eingelesen, so wird es mit der gespeicherten Parameterisierung neu aufgebaut.

2.1.8 Das Verzeichnis 'states'

In diesem Verzeichnis werden die Dateien abgelegt, in denen der jeweilige Ausführungs-Zustand der Modelle gespeichert ist (siehe dazu die Ausführungen zu den Befehlen 'store state', 'load state' und 'reset simulator' im Kapitel Bedienleisten).

.

2.1.9 Plattform-abhängige Verzeichnisse und Files

Zusätzlich zu den Standard-PACE-Dateien und -Verzeichnissen können noch weitere Dateien und Verzeichnisse vorliegen. Z.B. gibt es das Verzeichnis 'MAKEDLL', in dem die Dateien gespeichert sind, mit denen der Anwender eigene DLLs zur Erweiterung der virtuellen PACE-Maschine auf Knopfdruck erstellen kann.

2.2 Plattformspezifische Abhängigkeiten

Der Einsatz des Werkzeuges PACE ist weitgehend hardwareunabhängig. Das wenige, das beim Arbeiten mit PACE von der verwendeten Hardware-Plattform abhängig ist, wird in diesem Abschnitt beschrieben.

2.2.1 Fenster

PACE benutzt die Fenster des jeweiligen MS-Window-Systems. Die Art und Weise, wie ein PACE-Fenster verschoben, gerahmt, geschlossen und ikonisiert wird, ist Windows-Standard.

2.2.2 Farben

Die Farben, welche in PACE verwendet werden sind plattformabhängig. Durch Ausführen des Menüpunkts 'default platform colors' im 'view'-Menü der PACE-Hauptleiste können die Farben angezeigt werden, mit denen auf der aktuellen Plattform gearbeitet werden kann. Die Farbtabelle definiert die zur Verfügung stehenden Farben und deren Namen.

Mit weiteren Menüpunkten des ‘view’-Menüs können zahlreiche graphische Elemente der PACE-Netze und des jeweiligen Windows-Systems verändert werden.

2.2.3 Tastatur

Die deutsche Tastatur ist als Default-Tastatur definiert.

2.2.4 Maus

PACE benötigt eine Drei-Tasten-Maus, wie sie heute praktisch an jeder Anlage verfügbar ist. Die mittlere Maustaste ist durch die Druckfunktion des Selektionsrades gegeben.

3 DIE SPRACHE MSL

Die PACE-Systembeschreibungs-Sprache MSL (Modelling and Simulation Language) basiert auf der Netz-Theorie und auf Smalltalk. Attributierte Netze werden verwendet, um die System-struktur und den dynamischen Fluß der Objekte zu modellieren. Smalltalk wird benutzt, um die Datenstrukturen und deren Verarbei-tung zu beschreiben. Außerdem wird Smalltalk auch verwendet, um die Logik zu erweitern, nach welcher sich die Objekte innerhalb eines Netzes verhalten. Damit kann das Verhalten der Modelle datenabhängig gestaltet werden. In diesem Kapitel werden die Grundzüge von MSL beschrieben.

3.1 Attributierte Petri-Netze in MSL

Die in der Einleitung (Abschnitt 1.2) geforderte strukturelle Überein-stimmung von Realität und Modell lässt sich erreichen, wenn es die Modellierungsmethode ermöglicht, die visuelle Struktur einer Anwen-dung eins zu eins in ein Modell zu übertragen. Die Forderung nach übereinstimmender Wirkungsweise, d.h. nach funktionell identi-schem Verhalten, impliziert darüberhinaus, dass das Modell ausführ-bar, d.h. ein Simulationsmodell sein muss.

Die Analyse der für die exakte Modellierung von technischen und kommerziellen Anwendungen erforderlichen Beschreibungsmittel führte zur Entwicklung einer halbgrafischen Modellierungssprache MSL[3] . MSL bietet drei Arten von Sprachelementen:

---

[3] Modeling and Simulation Language

- **Sprachelemente zur Modellierung der Struktur einer Anwendung und insbesondere der Prozessbahnen, d.h. der Wege, auf denen die Prozesse stattfinden.**
  Die Prozesse werden dabei in Einzelschritte zerlegt, welche drei Aktionen durchführen:

  - Holen des Objekts von einem Lagerplatz (Speicher, Regal, Wartezimmer, usw.),
  - Verarbeiten des Objekts,
  - Ablegen des Objekts auf demselben oder einem anderen Lagerplatz.

  Die Feinheit der Einzelschritte bestimmt der Modell-Entwickler in Abhängigkeit vom gewünschten Detaillierungsgrad. Zu jedem Schritt werden die Objekttransformationen exakt niedergelegt.

- **Sprachelemente zur Modellierung der Objekte, die auf den Prozessbahnen bewegt werden.**
  Die Objekte werden durch die sie charakterisierenden Daten dargestellt. Bei den Operationen Holen und Ablegen werden Namen eingeführt, welche die Daten der anzusprechenden Objekte referenzieren.

- **Sprachelemente für die Beschreibung der Daten und Verarbeitungsalgorithmen.**
  Damit die Objekttransformationen in den modellierten Prozessen automatisch durchgeführt werden können, muss die Darstellung in mathematisch exakter Weise, beispielsweise in einer Programmiersprache, erfolgen. Die Verarbeitung verwendet die im vorangegangenen Punkt eingeführten Namen und hat die Aufgabe, die Datenbeschreibung der einlaufenden (abzuholenden) Objekte realitätsgerecht in die Datenbeschreibung der auslaufenden (abzulegenden) Objekte umzuwandeln.

Sprachelemente zur Modellierung und Visualisierung von Prozessbahnen kann man von den Petri-Netzen übernehmen. In MSL werden die Verarbeitungsschritte mit verallgemeinerten Stellen, Transitionen und Konnektoren beschrieben. Die Verallgemeinerung besteht darin, dass die Petri-Netz-Elemente mit zahlreichen

Attributen versehen werden, welche die genaue Verarbeitung der Objekte an den Netzknoten festlegen.[4]

Außer den aus S/T-Netzen bekannten Netzelementen enthält MSL zwei weitere Netzelemente für die hierarchische Strukturierung von Netzen, die als "Modul" und "Kanal" bezeichnet werden. Mit ihnen kann die Systemstruktur des realen Systems im Modell abgebildet werden. Module enthalten wiederverwendbare Teilnetze, die über Stellen und Kanäle als Schnittstellen in ein beliebiges Netz eingefügt werden können. Kanäle sind wie Stellen passive Netzelemente und werden als Zusammenfassung verwendet, wenn Module über mehrere Konnektoren verbunden sind.

Die zu bearbeitenden Objekte werden im Modell, ebenfalls in Anlehnung an Petri-Netze, durch Marken oder Behälter dargestellt, die auf den Prozessbahnen durch das Netz laufen und die im Modell benötigten charakteristischen Eigenschaften der realen Objekte als Attribute tragen. Damit die Objekte in den Transitionen ansprechbar sind, werden ihren Attributen durch Attributierung der Konnektoren sog. Konnektorvariablennamen zugeordnet. Die Bearbeitung (Veränderung) der Objekte und der sonstigen Prozessdaten wird hauptsächlich in den Transitionen vorgenommen, die Anweisungsfolgen in einer geeigneten Script- oder Programmiersprache enthalten. Derzeit wird in MSL für die Attributierung die benutzerfreundliche Programmiersprache „Smalltalk“ verwendet.

---

[4] Die Benennung der in PACE verwendeten erweiterten Petri-Netze als „attributierte Petri-Netze“ wurde von uns in Anlehnung an die Bezeichnung „attributierte Grammatiken“ gewählt, die bei der Entwicklung von Compilern eine wichtige Rolle spielen (siehe z.B. W.M.Waite und G. Goos: Compiler Construction (chapter 8: Attributed Grammars), Springer, ISBN 0-387-90821-8).

3.2 Petri-Netz-Erweiterungen

PACE-Modelle sind sog. High-Level Petri-Netze und gehören zu den graphischen Beschreibungs-Sprachen von ereignisorientierten, parallelen, diskreten Systemen. Es gibt zahlreiche Bücher und sehr viele wissenschaftliche Publikationen über Petri-Netze (siehe Literaturverzeichnis). Die wichtigsten Erweiterungen in PACE-Petri-Netzen sind:

- Attributierung von Marken
- Inskribierung von Transitionen und Konnektoren
- Hierarchiebildung
- Modellierung der Zeit
- Inhibierte Konnektoren
- Globale Variablen
- Zugang zu den Marken einer Stelle
- Kapazitätsbeschränkungen für Stellen
- Extra-Codes (MSL beschreibt „rechnende Petri-Netze").

3.3 Netz-Element-Typen

PACE-Netze sind sog. Stellen-Transitions-Netze. Sie bestehen aus zwei verschiedenen Knoten-Typen (S-Elemente und T-Elemente), aus Konnektoren und aus Marken.

3.3.1 S-Elemente

Die S-Elemente werden in der Regel durch Kreise dargestellt. Sie dienen der Zustandsdarstellung eines Modelles und bilden den passiven Teil der Netz-Elemente.

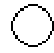
S-Elemente können durch Spezifikation ihrer Details verfeinert werden. Wir unterscheiden zwischen zwei verschiedenen S-Element-Typen:

- nicht verfeinerte S-Elemente werden Stellen genannt,
- verfeinerte S-Elemente werden Kanäle genannt.

Beide Typen werden im folgenden beschrieben.

3.3.1.1 Stellen

In PACE werden Stellen, wie in der Petri-Netz-Theorie üblich, durch einen weißen Kreis dargestellt. Stellen sind Behälter für die weiter unten beschriebenen Marken (siehe Abschnitt 3.3.5). Ihr Fassungsvermögen kann beschränkt werden, d.h. es kann festgelegt werden, wieviele Marken eine Stelle maximal aufnehmen darf. Außerdem kann die Anfangsbelegung einer Stelle definiert werden. Diese gibt an, welche Marken beim Start der Simulation vorliegen sollen. Stellen können mit einem Kommentar bzw. Titel versehen werden,



über den sie angesprochen werden.

Abb. 3-1: Standard-Darstellung einer Stelle

3.3.1.2 Kanäle

Kanäle bestehen aus einer Anzahl von S-Elementen (namentlich Stellen und Kanäle) und verbinden Module. Ein Kanal (Kanal-Unternetz) enthält keine lokalen T- Elemente (T-Elemente und Module sind weiter unten beschrieben).

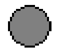


Abb. 3-2: Standard-Darstellung eines Kanals

Einerseits ist das Kanal-Konzept beim Arbeiten mit komplexen Netzen sehr nützlich, weil es die Verbindungen zwischen Moduln

zusammenfasst und damit die Übersicht erhöht, andererseits ist es aber nicht einfach, das Konzepz gleich von Beginn an vollständig zu verstehen und zu handhaben. Die folgende Analogie soll helfen, die Vorstellung, die hinter dem Kanal-Konzept steckt, besser zu verstehen:

Stellen verhalten sich wie Leitungen zwischen Hardware-Modulen. Diese können Untermodule beinhalten, also verschachtelt sein. Die einzelnen Einheiten der mit dem Modul verbundenen Leitung können zu je einem oder zu mehreren Untermodulen innerhalb des Hauptmoduls führen.

Kanäle verhalten sich wie Röhren, in denen Leitungen und andere Röhren zusammengefaßt werden. Letztere werden benutzt, um Verbindungen zu tiefer verschachtelten Untermodulen herzustellen. Die in einer Röhre eingeschlossenen Leitungen werden erst innerhalb des Zielmoduls einzeln angeschlossen. "Durchgangsmodule" werden von der Röhre nicht beeinflußt. Deshalb können Teile einer Leitung oder eines Rohres in verschiedenen Röhren vorkommen (ein S-Element in verschiedenen Kanälen).

Bestimmte T-Elemente, mit denen der Kanal im Ursprungsnetz verbunden ist, werden auch im Kanal-Unternetz dargestellt. Sie werden dabei mit geringerer Intensität gezeichnet und nur dann angezeigt, wenn sie von PACE automatisch mit einem S-Element innerhalb des Kanal-Unternetzes verbunden worden sind.

In einem Kanal-Unternetz wird ein Input-Konnektor, der ein S-Element mit einem T-Element verbindet, automatisch erstellt, wenn das S-Element in irgendeinem Unternetz des T-Elements als Input verwendet wird. Output-Konnektoren werden auf eine analoge Art und Weise erstellt. Anders ausgedrückt: ein Kanal-Unternetz zeigt, wie die darin enthaltenen Elemente innerhalb der T-Element-Hierarchie verwendet werden.

Ein Kanal-Unternetz kann auch nicht lokale S-Elemente beinhalten, welche aus einer höheren Hierarchiestufe importiert wurden. Diese Situation könnte so entstehen: angenommen das Unternetz eines Moduls enthält sowohl lokale als auch nicht-lokale S-Elemente.

Wenn nun Exemplare beider S-Elemente in einem Kanal zusammengefaßt werden, verlieren die nicht lokalen S-Elemente ihren Status nicht. Im neu gebildeten Kanal-Unternetz bleiben sie weiterhin nicht-lokal.

Nicht-lokale S-Elemente eines Modul-Unternetzes können auch in anderen Modul-Unternetzen und somit auch in verschiedenen Kanälen auf irgendeiner beliebigen Hierarchie-Ebene vorkommen.

3.3.2 T-Elemente

Die T-Elemente stellen den aktiven Teil eines Petri-Netzes dar. In ihnen werden die Marken verarbeitet und damit der Netzzustand geändert. T-Elemente sind mit zuführenden und mit abführenden S-Elementen verbunden. Zuführende S-Elemente werden auch Inputs (Eingaben), abführende S-Elemente Outputs (Ausgaben) der T-Elemente genannt. In PACE können auch die T-Elemente verfeinert werden. Nicht verfeinerte T-Elemente heißen Transitionen, verfeinerte heißen Module. Beide T-Element-Typen werden im folgenden beschrieben.

3.3.2.1 Transitionen

Eine Transition ist ein nicht verfeinertes T-Element. In PACE werden Transitionen standardmäßig durch einen grauen Balken dargestellt. Transitionen verändern beim Feuern den Zustand ihrer Nachbarstellen, dargestellt durch die darauf liegenden Marken. Sie ziehen Marken von ihren Input-Stellen ab und erzeugen neue Marken auf ihren Output-Stellen.



Abb. 3-3: Default- und Alternativ-Ikone für Transitionen

3.3.2.2 Module

Ein Modul ist ein verfeinertes T-Element. In PACE werden Module standardmäßig durch ein graues Quadrat dargestellt. Ihre Verfeinerung kann aus allen Elementen des PACE-Sprachumfanges, im besonderen auch aus anderen Modulen, bestehen. Zu jedem Modul gehört ein Unternetz, das die Verfeinerung beschreibt, und zu dem der Modul die Schnittstelle bildet. Es kann mit einem Namen versehen werden. Jeder Modul kann einzeln gespeichert und wieder geladen werden. Dabei wird sein Name als Filename benutzt. Daher muß der Name eines Moduls den Konventionen für Filenamen des jeweiligen Betriebssystems entsprechen.

Ein S-Element, das mit einem Modul verbunden ist, wird in der Regel auch innerhalb des entsprechenden Modul-Unternetzes (schwächer gezeichnet) dargestellt. Innerhalb dieses Unternetzes kann es wiederum mit anderen T-Elementen (Transitionen und Moduln) verbunden werden.

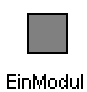


Abb. 3-4: Standard-Darstellung eines Moduls

3.3.3 Kommentarboxen

Kommentarboxen können überall in einem Netz angebracht werden. Sie können einen beliebigen Text enthalten und sind weder auf ein einzelnes Element bezogen, noch haben sie irgend einen Einfluß auf das Verhalten des Netzes. Kommentarboxen werden lediglich zu Dokumentationszwecken benutzt.

3.3.4 Konnektoren

Konnektoren verbinden S- und T-Elemente. Sie werden durch Pfeile dargestellt. Die Pfeilrichtung wird immer in Bezug zu den T-Elementen bezeichnet. Konnektoren, die von einem S-Element zu einem T-Element führen, werden Input-Konnektoren genannt. Output-Konnektoren führen von einem T-Element zu einem S-Element.

PACE unterscheidet zwischen drei Konnektor-Typen:

3.3.4.1 T-Konnektoren

Ein T-Konnektor verbindet eine Transition mit einer Stelle und kann mit einer Inskription versehen werden, welche die Marken, die über den Konnektor fließen dürfen, einschränkt. Die folgende Abbildung zeigt ein Beispiel eines T-Konnektors mit einer Inskription, die aus zwei Elementen besteht (der Zahl 2 und der Konnektorvariablen x).



Abb. 3-5: T-Konnektor mit Inskription

3.3.4.2 M-Konnektoren

Als M-Konnektor wird ein Konnektor bezeichnet, der zwei Elemente miteinander verbindet, von denen mindestens eines ein verfeinertes Element ist. M-Konnektoren können nicht inskribiert werden.

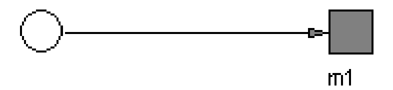


Abb. 3-6: M-Konnektor

3.3.4.3 Inhibitoren

Ein Inhibitor ist ein spezieller Input-Konnektor, der durch einen kleinen Kreis auf seiner Pfeilspitze gekennzeichnet ist. Dieser

Konnektortyp wird benutzt, um das Feuern einer Transition zu verhindern.
Das Inhibieren wird durch die Inhibitor-Inskription geregelt. Bei Übereinstimmung von Inhibitor-Inskription und der Inskription einer Marke auf einer durch einen Inhibitor mit einer Transition verbundenen Stelle, wird diese Marke das Feuern der Transition verhindern. Der Inhibitor kann als Barriere betrachtet werden: Keine Marke mit der Inhibitor-Inskription wird jemals darüber fließen können.

Die Transition, die in der folgenden Abbildung dargestellt ist, kann nur feuern, wenn auf der zuführenden Stelle keine Marke mit der Integer-Zahl 2 als Attribut vorliegt.

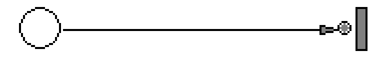


Abb. 3-7: Inhibierter Input-Konnektor

3.3.5 Marken

Die Marken stellen die dynamischen Elemente dar, die sich innerhalb des Netzes bewegen. In der Regel werden sie in PACE, wie in der Petri-Netz-Theorie üblich, durch schwarze Punkte dargestellt. Jeder einzelnen Marke kann aber, wie auch den statischen Netzelementen, ein beliebiges Bild (Ikone) zugewiesen werden, mit der das Objekt, welches durch die Marke repräsentiert wird, visualisiert werden kann. Marken werden von den Input-Stellen einer Transition abgezogen und von ihr verbraucht. Danach werden gemäß den Benutzerangaben und Inskriptionen für die Transition neue Marken erzeugt, die auf die Output-Stellen der Transition gelegt werden. Damit dieser Vorgang, d.h. der Ablauf des Netzes, animierbar ist, werden die Marken während der Simulation nicht unmittelbar ausgetauscht, sondern wandern mit wählbarer Geschwindigkeit von den Eingabestellen zur Transition und von dort zu den Ausgabestellen.

Eine Marke kann eine beliebige Anzahl von Attributen tragen. Diese werden bei der Bildschirmanzeige der Marken in runden Klammern dargestellt. Jedes Attribut ist ein Smalltalk-Objekt und kann erzeugt werden, indem entweder das Netz mit seiner definierten Anfangsbelegung neu initialisiert wird oder indem eine Transition durch Feuern eine neue Marke erzeugt. Die Smalltalk-Objekte werden mit den entsprechenden Attributen wieder freigegeben, wenn die entsprechende Marke oder das entsprechende Attribut von einer Transition verbraucht wurde.

Nach dem Verbrauchen und Erzeugen von Marken werden diese auf die Output-Stellen weitergeleitet. In der folgenden Abbildung ist eine attributierte Marke (Attribut ist die Integer-Zahl 22) dargestellt, die sich auf einer Stelle befindet. Eine zweite Marke mit dem gleichen Attribut fließt gerade über den Konnektor.

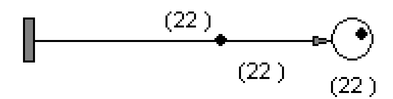


Abb. 3-8: Marke mit dem Attribut '22'

3.4 Netz-Hierarchiestruktur

Mit PACE können hierarchisch geordnete System-Modelle erstellt werden. Ihre Einbettung in eine vorgegebene Umgebung (bestehende Netze oder Interface zur Außenwelt) und die Bildung einer Netz-Hierarchiestruktur kann bis in die tiefsten Verarbeitungsebenen detailliert beschrieben werden.

Um ein sehr verzweigtes Netz übersichtlich zu halten, sollten die einzelnen Elementgruppen in hierarchisch strukturierte Einheiten zusammengefaßt werden. Eine solche Hierarchiestruktur entsteht, indem bestimmte T-Elemente zu Modulen zusammengefasst und weiter verfeinert werden.

Auch S-Elemente können als Kanäle vereinbart und weiter verfeinert werden. Diese Art der Verfeinerung beschreibt aber nicht direkt die Hierarchiestruktur eines Modelles. Sie strukturiert lediglich das Verhalten einer komplexen Kommunikation zwischen Unternetzen, da Kanäle (verfeinerte S-Elemente) verschiedene S-Elemente zusammenfassen.

Die Netze der oberen Hierarchiestufen bestehen aus Modulen und S-Elementen, die mit Konnektoren miteinander verbunden sind. In diesen Netzen und in der Modul-Liste spiegelt sich die Hierarchiestruktur des spezifizierten System-Modells wider. Die oberste Ebene enthält meist das zu programmierende System und dessen Umwelt-Simulation. Ebenso kommen darin die Kommunikationskanäle zwischen dem System und der Umwelt vor. Das exakte Verhalten des Netzes wird in den untersten Hierarchiestufen beschrieben, gewissermaßen in den Blättern des Hierarchie-Baumes der Module. Im Unternetz eines Moduls wird ein Teil des gesamten System-Modelles beschrieben. Das Unternetz eines Kanals hingegen ist eine Sammlung von Verbindungen zwischen Modulen.

Ein bestimmtes S- oder T-Element kann in einer beliebigen Anzahl von Unternetzen vorkommen. In den Netzen der unteren Hierarchiestufen werden die Standard-Ikonen solcher Netz-Elementen mit

hellerem Rand beziehungsweise mit hellerem Inneren dargestellt. Diese können innerhalb eines Fensters, das ein Netz der unteren Hierarchiestufe darstellt, beliebig positioniert werden. In allen Ebenen stehen auch die meisten Funktionen zur Verfügung, um dieses Element zu bearbeiten. Die Lösch-Funktion kann aber ausschließlich in dem Teil-Netz ausgeführt werden, in dem sich das betreffende Element befindet (bzw. in dem es erstellt wurde).

Ein S-Element, das im Netz eines verfeinerten T-Elementes vorkommt, kann mit dessen T-Elementen verbunden werden. Innerhalb des S-Element-Unternetzes können hingegen keine Verbindungen spezifiziert werden. Es werden aber die Verbindungen angezeigt, die sich vom hierarchisch eine Stufe höher gelegenem Netz ergeben. Die Konstruktion des System-Modelles erfolgt innerhalb der Hierarchie der T-Element-Module.

Oft ergibt sich eine natürliche Gliederung. Ein Modell einer Firma oder einer Bank könnte beispielsweise wie folgt gegliedert sein:

Oberste Ebene: Direktion
1. Unterebene: Hauptabteilungen
2. Unterebene: Abteilungen
3. Unterebene: Arbeitsgruppen
4. Unterebene: Einzelne Arbeitsplätze.

3.4.1 Wie wird ein System strukturiert?

Ein System-Modell sollte in aktive Einheiten, in denen Zustandsänderungen stattfinden, gegliedert werden. Jede einzelne aktive Einheit wird durch einen Modul dargestellt und im zugehörigen Netz weiter verfeinert. Diese Module werden durch S-Elemente (je nach Kontext sind das Stellen oder Kanäle) miteinander verbunden. Über die Konnektoren zwischen den Netzelementen fließen die zu verarbeitenden Objekte. In PACE geht ein solcher Fluß immer über ein S-Element.

Wenn verschiedene S-Elemente innerhalb des gleichen Moduls zum Einsatz kommen, können diese zu einem Kanal zusammengefaßt werden. Auf der oberen Hierarchiestufe wird dann nur die

Zusammenfassung in Form der Ikone des Kanals dargestellt, wodurch das Netz erheblich übersichtlicher wird. In der Darstellung des Netzes der weiter unten liegenden Hierarchiestufe, in dem auch die eigentliche Objektverarbeitung stattfindet, werden die im Kanal zusammengefaßten S-Elemente einzeln angezeigt.

Normalerweise sollte jedes Netzfenster, der Übersichtlichkeit halber, höchstens 10 bis 20 Elemente aufweisen. Werden mehr benötigt, so ist es ratsam, das Netz aufzuteilen.

3.4.2 Konsistenz-Regeln

PACE überprüft interaktiv die Konsistenz eines Modells. Innerhalb des Unternetzes eines Moduls werden auch die S-Elemente angezeigt, die mit dem Modul verbunden sind. Damit diese als S-Elemente einer oberen Hierarchiestufe erkannt werden können, stellt sie PACE mit geringerer Intensität dar. Diese S-Elemente können mit den T-Elementen, die Teil des Unternetzes sind, verbunden werden. Es sollte aber beachtet werden, daß Stellen, die im Ursprungsmodul zuführend waren, auch im Unternetz nur wieder zuführende Input-Stellen sein können. Dasselbe gilt natürlich auch für Output-Stellen.

Wird ein Kanal in ein Modul-Unternetz integriert, und werden die in ihm zusammengefaßten Elemente sichtbar gemacht, gelten diese Konsistenz-Regeln auch für die Teilelemente des Kanals.

3.4.3 Verbindungen innerhalb der Kanal-Unternetze

In der Beschreibung des S-Element-Typs "Kanal" wird erklärt, wie die Verbindungen und die T-Elemente innerhalb des Kanal-Unternetzes sichtbar gemacht werden können (siehe Abschnitt 3.3.1).

3.5 Smalltalk-Netz-Inskriptionen

Die meisten Elemente in einem PACE-Netz können mit Text (Kommentare oder Smalltalk-Inskriptionen) versehen werden. Während Kommentartexte als freier Fließtext verfaßt werden, bestehen die eigentlichen Inskriptionen aus Smalltalk-Anweisungen und müssen daher den Smalltalk-Konventionen genügen.

Das Verhalten von Transitionen beim Feuern wird über Inskriptionen festgelegt. Attribute von Marken, die über die einzelnen Netz-Elemente fließen, können mit Smalltalk-Anweisungen verändert werden.

Diese Interaktion findet statt, sobald Marken von Transitionen verbraucht werden, die von der Marke angelieferten Objekte (Input-Variablen) also für Berechnungen bereitstehen. Die Botschaften, aus denen eine Inskription besteht, können diese Objekte verändern oder ihren Wert testen, um zu entscheiden, ob die Transition feuern darf. Beim Feuern einer Transition werden die neu berechneten Objekte gemäß den Angaben in den Inskriptionen den Output-Marken zugewiesen (Output-Variable).

3.5.1 Attribute der Initial-Marken

Jedem Attribut einer Initial-Marke kann Smalltalk-Code zugeordnet werden. Mit diesem Code werden die Eigenschaften der Marke festgelegt. Werden mehreren Smalltalk-Anweisungen angegeben, so wird der Wert der letzten Anweisung als Ergebnis zurückgeliefert. Der inskribierte Smalltalk-Code kann aus einem ganz einfachen literalen Objekt, beispielsweise aus einer Ganzzahl oder einem Symbol, bestehen. Er kann aber auch irgendein komplexes Programm sein. In Abb. 3-9 werden für eine Stelle vier Initial-Marken vereinbart. Das zweite Attribut der dritten Initialmarke ist das Codestück: „Uniform from: 2.0 to: 100”. Der Initialmarke wurde ein Anfangsikon zugeordnet, das im rechten Fenster angezeigt wird und

die Marke bei einer Animation von der Stelle bis zur ersten Transition repräsentiert.

Der Code, der die Attribute der Initial-Marken generiert, wird jedesmal ausgeführt, wenn ein Modell neu initialisiert wird.

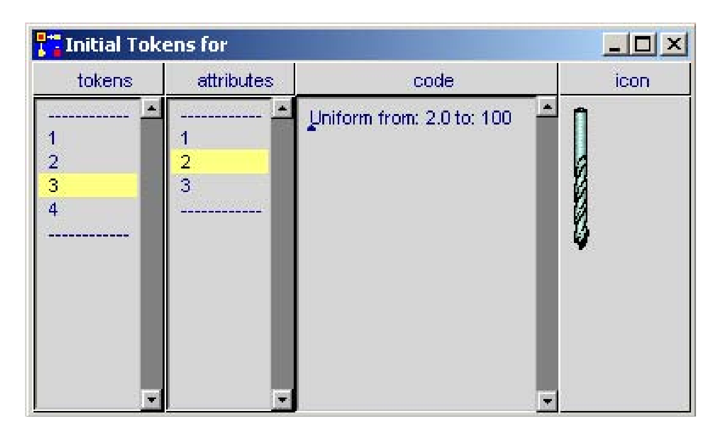


Abb. 3-9: Marken-Listen-Fenster für eine Stelle

3.5.2 Transitions-Codes

Ein Transitions-Code besteht aus drei Teilen: Bedingung (condition), Verzögerung (delay) und Aktion (action). Jeder dieser drei Code-Teile besteht entweder aus einer Anzahl Smalltalk-Anweisungen oder ist leer. Sowohl die Input-, Output- als auch die temporären Variablen einer Transition können in allen drei Code-Teilen verwendet werden.

Um sofort erkennen zu können, um welchen Transitioncode es sich jeweils handelt, werden die verschiedenen Transitioncodes ab PACE ‚Version 5 in Farbe dargestellt.[5]

[5] In einem Teil der Abbildungen der PACE-Dokumentation, die noch mit früheren PACE-Versionen erstellt wurden, sind die verschiedenen Transitioncodes noch in schwarzer Farbe dargestellt.

3.5.2.1 Bedingungs-Code (Condition)

Das Ergebnis eines Bedingungs-Codes muß ein boolesches Objekte sein (true oder false). Als Default-Wert wird 'true' angenommen.

Mit dem Bedingungs-Code kann das Aktivieren einer Transition von den Werten der Input-Variablen abhängig gemacht werden. Ergibt die Abarbeitung des Bedingung-Codes den Wert 'false', so kann die Transition nicht feuern. Jede Auswertung des Bedingungs-Codes wird für eine bestimmte Kombination von Input-Marken durchgeführt. Diese Marken bestimmen die Werte der Input-Variablen.

In dem folgenden Beispiel (Abbildung 3-10) akzeptiert der Bedingungs-Code nur die Marke mit dem Attribut '7' auf der Stelle 'p1' für ein gemeinsames Feuern mit einer der zwei Marken, die sich auf der Stelle 'p2' befinden.

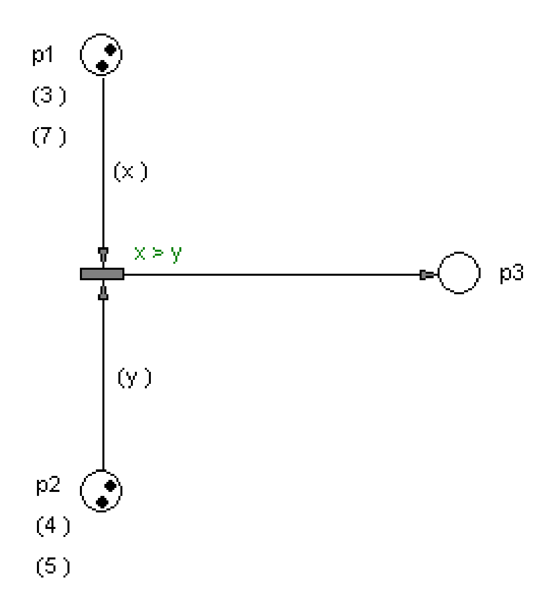


Abb. 3-10:Beispiel für einen Bedingungs-Code

Zu beachten ist, daß der Bedingungscode nur bei der Änderung des Zustands einer Input-Stelle einer Transition überprüft wird. Auch wenn durch spätere Maßnahmen in einem anderen Teil des Netzes die Auswertung des Bedingungscodes zu dem Wert ‘true’ führen würde, so wird die betroffene Transition nicht automatisch feuern. Die wiederholte Auswertung des Bedingungscodes muß bei der Modellierung organisiert werden, indem beispielsweise eine Marke periodisch in einer Schleife eine der Input-Stellen der betroffenen Transition passiert.

3.5.2.2 Verzögerungs-Code (Delay)

Dem Verzögerungs-Code können nur nicht-negative Zahlen (oder 'nil') als Werte zugewiesen werden. Ist der Code leer (oder ist sein Wert 'nil'), so wird die Transition unmittelbar nach Überprüfung ihrer Bedingungen gefeuert. Andernfalls wird die Feuerung entsprechend den im Verzögerungs-Code angegebenen Zeiteinheiten verzögert.

3.5.2.3 Aktions-Code (Action)

Der Aktions-Code wird abgearbeitet und ausgewertet, sobald die Transition feuert, das heißt, sobald die durch den Bedingungscode und/oder den Verzögerungscode bestimmte Verzögerung verstrichen ist.

Der Aktions-Code wird in der Regel eingesetzt, um den Output-Variablen Werte zuzuweisen, um Attribute zu verändern, oder um parallele Aktivitäten zu veranlassen (z.B. um eine Meldung auf den Bildschirm bringen, eine Benutzerfunktion aufzurufen, eine globale Variable zu verändern, einen Wert von der Prozeßperipherie einzulesen, usw.).

3.5.3 Die Pseudo-Variable 'self'

'self' ist eine sogenannte Pseudo-Variable. In PACE wird 'self' ausschließlich in Inskriptionen eingesetzt und hat hier die Funktion einer Code-Verankerung. Diese Code-Verankerung unterstützt einige wenige, PACE-spezifische Methoden, die an verschiedenen

Stellen dieses Handbuchs beschrieben sind. Die Methoden werden aufgerufen, indem an die Pseudo-Variable 'self' Botschaften gesandt werden.

Einige wichtige Methoden, die in Transitionscodes verwendbar sind, werden im folgenden beschrieben.

3.5.3.1 Zugang zur Belegung einer Stelle

Eine Referenz auf die Belegung einer bestimmten Stelle ist von einem Transitions-Code aus zugänglich, indem folgende Botschaft gesandt wird:

(self placeNamed: 'Name der Stelle') marking

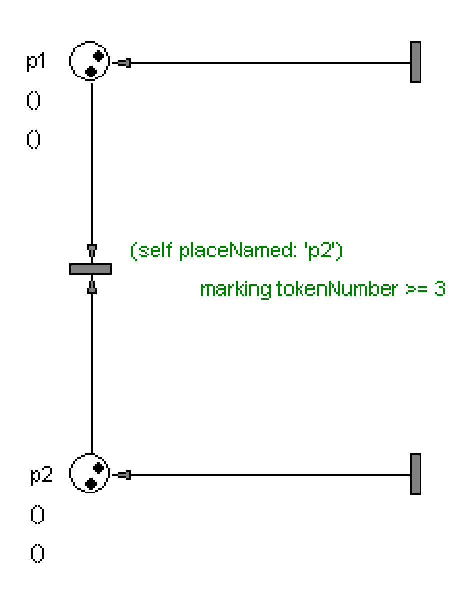


Abb. 3-11:Beispiel für den Zugang zur Belegung einer Stelle

Ausgehend von dieser Referenz kann man über weitere Botschaften auf die Eigenschaften von Marken zugreifen. Diese Botschaften sind

weiter unten im Unterkapitel 'Zugang zu den Marken einer Stelle' beschrieben.

Zu beachten ist, daß die Berechnung der Referenz für Stellen, die nicht mit der aktiven Transition verbunden sind, viel Rechenzeit benötigt und deshalb nicht in Inskriptionen verwendet werden sollte, die häufig durchlaufen werden. In solchen Fällen ist es zweckmäßig, die Referenz in einer Inskription (z.B. im Initialisierungscode), die selten durchlaufen wird, zu berechnen und z.B. einer globalen Variable oder einer Modulvariablen zuzuweisen (siehe Abschnitt 3.7.1).

In Abbildung 3-11 wird die Transition, die sich in der Mitte befindet, nur dann aktiviert, wenn sich mindestens drei Marken auf der Stelle 'p2' befinden.

3.5.3.2 Beenden einer Simulation

Wird die Anweisung:

self terminate

in einer Transitions-Inskription ausgeführt, so wird die Ausführung des Netzes nach der vollständigen Ausführung der gerade in Bearbeitung befindlichen Inskription mit dem ‘terminate’-Befehl beendet. Der Termination-Code des Modells wird ausgeführt. Wurde eine Simulation auf diese Weise zum Stillstand gebracht, so kann sie durch Ausführen des Menüpunkts 'initialize' im Simulator-Menü wieder neu initialisiert werden.

3.5.3.3 Anhalten eines Simulationslaufs

Ein Simulationslauf kann durch den Aufruf:

self break

angehalten werden. Ein ggf. vorgesehener Stop-Code wird nach dem Anhalten ausgeführt.

Der Simulationslauf kann vom Anwender über das Simulator-Menü (No-Selection-Menu in einem Netzfenster, das sich im Simulations-modus befindet) oder mit dem ‘continue’-Knopf einer Exekutive fortgesetzt werden.

3.5.3.4 Programm-gesteuertes Wiederstarten einer Simulation

Die Anweisung:

self restart

bewirkt, daß nach der Ausführung der gesamten Inskription, welche den ‘restart’-Befehl enthält, zunächst der Termination-Code des Modells und danach sein Initialisierungs-Code ausgeführt wird. Danach wird das Netz automatisch neu gestartet.

Diese Anweisung ist für die Durchführung von Optimierungsläufen wichtig (siehe auch das PACE-Handbuch: Optimierung).

Mit der Anweisung

self isRestarted

kann während des Programmablaufs festgestellt werden, ob man sich im ersten Durchlauf des Modell befindet. Diese Anweisung liefert einen der booleschen Werte true oder false und wird z.B. benötigt, wenn bestimmte Voreinstellungen nur zu Beginn des ersten

Durchlaufs durchgeführt werden sollen (z.B. Initialisieren vor dem ersten Durchlauf und danach Zählen der Durchläufe).

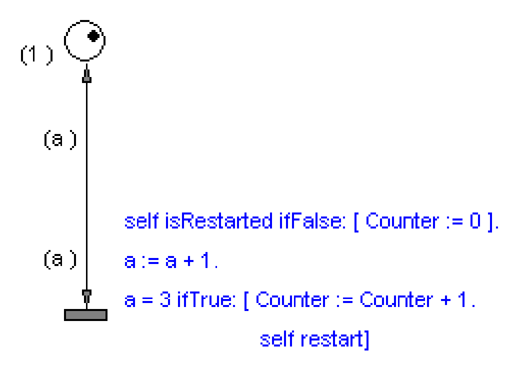


Abb. 3-12: Beispiel für die Verwendung von ‘restart’ und ‘isRestarted’

3.5.3.5 Ausführungsmodus eines Simulationslaufs feststellen

Mit der Methode:

self inAnimationMode

kann man feststellen, ob ein Simulationslauf im Animations- oder im Hintergrund-Modus ausgeführt wird. Die Methode liefert den Wert **true** wenn ein Animationslauf durchgeführt wird, andernfalls den Wert **false**.

3.5.3.6 Ändern der Ikone eines Knotens

Während der Simulation kann die Ikone jedes Knoten geändert werden. Dazu sendet der Transitions-Code dem Knoten die folgende Botschaft:

setIconNamed:

(Beispiel siehe weiter unten).

Das Argument dieser Botschaft ist der Name der neu zu verwenden-den Ikone. Dabei muß beachtet werden, daß die Ikonennamen Smalltalk-'Symbole', die Knotennamen hingegen Smalltalk-'Strings' sind.

Der Hinweis auf den Knoten, dem die Botschaft gesandt werden soll, wird vom Transitions-Code aus wie folgt ermittelt:

Stellen:

self placeNamed: ‘naming comment'

Module:

self moduleNamed: 'modulname'

Transitionen:

self transitionNamed: ‘naming comment’

Beispiel:

Die folgende Botschaft veranlaßt, daß anstelle der aktuellen Ikone des T-Elements mit Namen 'm1' die Ikone '#icon1' dargestellt wird:

(self moduleNamed: 'm1') setIconNamed: #icon1

3.5.3.7 Selektieren von Szenen

Szenen (siehe Kapitel Bedienleisten, Abschnitt Support-Menü) können auch von Inscriptionen her selektiert und deselektiert werden. Beispielsweise kann man beim Anhalten der Simulation über den Stop-Code eine bestimmte Szene mit weitergehender Information oder mit Eingabefenstern hochbringen und diese dann bei der Fortsetzung des Simulationslaufs über den Continue-Code

wieder verbergen. Im Continue-Code können außerdem zwischenzeitlich geänderte Parameter eingelesen werden.

Die folgenden beiden Methoden stehen zur Verfügung:

· **hideScenery**

Die Methode verbirgt die gerade angezeigte Szene. Ist keine Szene angezeigt, so ist die Methode wirkungslos.

Beispiel: self hideScenery.

· **showScenery:**

Die Methode zeigt die Szene an, deren Bezeichner als Argument (in Form eines String) angegeben ist. Gibt es die angegebene Szene nicht, so hat die Methode keine Wirkung.

Beispiel: self showScenery: 'Scene1'.

3.5.3.8 Initialisierung der Belegung eines Moduls

Die folgende Botschaft initialisiert die Initialmarken aller Stellen, die sich innerhalb eines Modul-Unternetzes (in unserem Fall 'm1' genannt) befinden:

(self moduleNamed: 'm1') setInitialMarking .

Diese Botschaft kann von einem Transitions-Code des Modul-Unternetzes gesandt werden. Die Transition mit diesem Code muß keine Input-Marken verbrauchen, weil diese Marken zerstört und anschließend durch die Initialmarken ersetzt werden.

3.5.3.9 Ausnahmebehandlung

Tritt während des Ausführens eines Transitions-Codes ein Laufzeitfehler auf, so erscheint auf dem Bildschirm das in Abb. 3-13 abgebildete Fehlerfenster. Die erste Zeile des Fensters beschreibt das Problem, das zu dem Fehler geführt hat. Die zweite Zeile gibt den Anfang der Transitions-Inskription wieder, die gerade ausgeführt wurde. Der untere Teil dieses Fensters besteht aus der Frage "Open

net window?" (Netz-Fenster öffnen?) und den Schaltern "yes" und "no". Durch Anklicken des Schalters "yes" wird ein Fenster geöffnet, in dem das Netz angezeigt wird, das die Transition beinhaltet, die zum Fehler geführt hat. Der Cursor zeigt auf das Netzelement, in dem der Fehler festgestellt wurde. Auf diese Weise kann die fehlerhafte Programmstelle schnell lokalisiert und korrigiert werden. Der Ablauf der Simulation wird in jedem Fall unterbrochen.



Abb. 3-13: Fenster mit der Meldung eines Laufzeitfehlers

3.5.3.10 Bestimmung der eigenen Position

Die Position einer Transition im Fenster wird mit der Botschaft:

self getMyPosition

als Punkt x@y angeliefert.

Die x- und die y-Richtung des Fensters ist jeweils in 1000 Einheiten aufgeteilt. Die Botschaft liefert Koordinaten x und y zwischen 0 und 1000, welche der Ort der Transition in diesem Raster angeben.

3.5.3.11 Position einer Transition im Netzfenster

Die Position einer Transition im Netzfenster wird mit der Botschaft:

self getTransitionPosition: transition-name-as-string

als Punkt x@y angeliefert. transition-name-as-string ist der Kommentar der Transition (normalerweise deren Name). Der

Anwender muß darauf achten, daß der angegebene transition-name-as-string eindeutig ist, d.h. nur bei einer Transition auftritt.

Die x- und die y-Richtung des Netzfensters ist jeweils in 1000 Einheiten aufgeteilt. Die Botschaft liefert Koordinaten x und y zwischen 0 und 1000, welche der Ort der Transition in diesem Raster angeben.

Die Methode kann auch im Modell-Code verwendet werden.

3.5.3.12 Position einer Stelle im Netzfenster

Die Position einer Stelle im Netzfenster wird mit der Botschaft:

self getPlacePosition: place-name-as-string

als Punkt x@y angeliefert. place-name-as-string ist der Kommentar der Stelle (normalerweise deren Name). Der Anwender muß darauf achten, daß der angegebene place-name-as-string eindeutig ist, d.h. nur bei einer Stelle auftritt.

Die x- und die y-Richtung des Netzfensters ist jeweils in 1000 Einheiten aufgeteilt. Die Botschaft liefert Koordinaten x und y zwischen 0 und 1000, welche der Ort der Stelle in diesem Raster angeben.

Die Methode kann auch im Modell-Code verwendet werden.

3.5.3.13 Position eines Kanals im Netzfenster

Die Position eines Kanals im Netzfenster wird mit der Botschaft:

self getChannelPosition: channel-name-as-string

als Punkt x@y angeliefert. channel-name-as-string ist der Kommentar der Stelle (normalerweise deren Name). Der Anwender muß darauf achten, daß der angegebene channel-name-as-string eindeutig ist, d.h. nur bei einem Kanals auftritt.

Die x- und die y-Richtung des Netzfensters ist jeweils in 1000 Einheiten aufgeteilt. Die Botschaft liefert Koordinaten x und y zwischen 0 und 1000, welche der Ort des Kanals in diesem Raster angeben.

Die Methode kann auch im Modell-Code verwendet werden.

3.5.3.14 Position eines Moduls im Netzfenster

Die Position eines Moduls im Netzfenster wird mit der Botschaft:

self getModulePosition: module-name-as-string

als Punkt x@y angeliefert. module-name-as-string ist der Kommentar der Stelle (normalerweise deren Name). Der Anwender muß darauf achten, daß der angegebene module-name-as-string eindeutig ist, d.h. nur bei einem Modul auftritt.

Die x- und die y-Richtung des Netzfensters ist jeweils in 1000 Einheiten aufgeteilt. Die Botschaft liefert Koordinaten x und y zwischen 0 und 1000, welche der Ort des Moduls in diesem Raster angeben.

Die Methode kann auch im Modell-Code verwendet werden.

3.5.3.15 Positionieren einer Stelle

Die Position einer Stelle im Netzfenster wird mit der Botschaft:

self setPlace: place-name-as-string position: aPoint

place-name-as-string ist der Kommentar der Stelle (normalerweise deren Name). aPoint ist ein Punkt x@y. Der Anwender muß darauf achten, daß der angegebene place-name-as-string eindeutig ist, d.h. nur bei einer Stelle auftritt.

Die x- und die y-Richtung des Netzfensters ist jeweils in 1000 Einheiten aufgeteilt. Die Botschaft verlangt Koordinaten x und y des Punkts zwischen 0 und 1000, welche der Ort der Stelle in diesem Raster angeben.

Die Methode kann auch in einem Modell-Code verwendet werden.

3.5.3.16 Positionieren einer Transition

Die Position einer Transition im Netzfenster wird mit der Botschaft:

self setTransition: transition-name-as-string position: aPoint

transition-name-as-string ist der Kommentar der Transition (normalerweise deren Name). aPoint ist ein Punkt x@y. Der Anwender muß darauf achten, daß der angegebene transition-name-as-string eindeutig ist, d.h. nur bei einer Stelle auftritt.

Die x- und die y-Richtung des Netzfensters ist jeweils in 1000 Einheiten aufgeteilt. Die Botschaft verlangt Koordinaten x und y des Punkts zwischen 0 und 1000, welche der Ort der Transition in diesem Raster angeben.

Die Methode kann auch in einem Modell-Code verwendet werden.

3.5.3.17 Positionieren eines Kanals

Die Position eines Kanals im Netzfenster wird mit der Botschaft:

self setChannel: channel-name-as-string position: aPoint

channel-name-as-string ist der Names des Kanals. aPoint ist ein Punkt x@y. Der Anwender muß darauf achten, daß der angegebene channel-name-as-string eindeutig ist, d.h. nur bei einem Kanal auftritt.

Die x- und die y-Richtung des Netzfensters ist jeweils in 1000 Einheiten aufgeteilt. Die Botschaft verlangt Koordinaten x und y des Punkts zwischen 0 und 1000, welche der Ort des Kanals in diesem Raster angeben.

Die Methode kann auch in einem Modell-Code verwendet werden.

3.5.3.18 Positionieren eines Moduls

Die Position eines Moduls im Netzfenster wird mit der Botschaft:

self setModule: module-name-as-string position: aPoint

module-name-as-string ist der Names des Kanals. aPoint ist ein Punkt x@y. Der Anwender muß darauf achten, daß der angegebene module-name-as-string eindeutig ist, d.h. nur bei einem Modul auftritt.

Die x- und die y-Richtung des Netzfensters ist jeweils in 1000 Einheiten aufgeteilt. Die Botschaft verlangt Koordinaten x und y des Punkts zwischen 0 und 1000, welche der Ort des Moduls in diesem Raster angeben.

Die Methode kann auch in einem Modell-Code verwendet werden.

3.5.3.19 Hintergrundfarbe hinter Netzelementen

Die Hintergundfarbe hinter einem Netzelement kann mit den folgenden Methoden bestimmt werden:

self colorAtChannel: a-channel-name-as-string
self colorAtModule: a-module-name-as-string
self colorAtPlace: a-place-name-as-string
self colorAtTransition: a-transition-name-as-string

Die Farbe wird als Array mit 4 Elementen angeliefert. Das erste Element bezeichnet die Art der Farbdarstellung und ist das Smalltalk-Symbol #RGB. Die nächsten drei Elemente beschreiben der Reihe nach den Rot-, Grün- und Blauanteil. Z.B. wird die Farbe cyan durch #(#RGB 0.0 1.0 1.0) dargestellt.

Beispiel: self colorAtModule: 'Einheit'.

3.5.3.20 Name des Netzes

Der Name des Netzes (identisch mit dem Namen des Supermoduls), in dem sich eine Transition befindet, wird mit der Methode:

self isInModule

bestimmt. Ergebnis ist der Name des Netzes als String.

3.5.3.21 Interface eines Moduls

Die mit einem Modul verbundenen Stellen können mit der Methode

self getConnectedSElements: module-name-as-string

bestimmt werden. Die Methode liefert eine OrderedCollection, deren Elemente Associationen sind. Jede Association besteht aus dem Namen einer mit dem Modul verbundenen Stelle (key) und einer der Angabe #InputConnector oder #OutputConnector (value).

3.5.3.22 Hintergrundbild einsetzen

Eine der gespeicherten Ikonen kann als Hintergundbild einem Netz (Modul) zugeordnet werden und wird dann beim Öffnen des zugeordneten Fensters angezeigt. Das Einsetzen der Ikone erfolgt mit der Methode:

self insertBackgroundImage: an-icon-name-as-string
module: a-module-name-as-string

Diese Methode kann nur verwendet werden, wenn der betreffende Modul Untermodule besitzt.

3.5.3.23 Unerwünschte Seiteneffekte

Unerwünschte Seiteneffekte können z.B. auftreten, wenn die Bedingungs-Codes in den Transitionen Codeteile enthalten, die eigentlich in einem Action-Teil angegeben werden sollten. Ein Bedingungs-Code könnte in einem solchen Fall die Attribute der Input-Marken verändern und dann 'false' als Rückgabewert haben. Die Attribute der Input-Marken sollten nur im Aktion-Code einer Transition verändert werden!

3.5.4 Konnektor-Attribute

Die Konnektoren lassen sich mit einer Attributbeschreibungs-Liste (Konnektor-Inskriptionen) versehen. Bei T-Konnektoren, die keine Inskriptionen tragen, wird eine leere Attributliste angenommen. Attributbeschreibungen können entweder Variablen oder konstante Literal-Objekte sein.

Eine Marke kann erst dann über einen Input-Konnektor fließen, wenn all ihre Attribute mit denjenigen des Konnektors übereinstimmen. Zuerst überprüft der Auswahlalgorithmus die Anzahl der Attribute. Stimmt diese, wird danach die Art der Attribute überprüft.

Beim Übereinstimmen von konstanten Attributbeschreibungen muß eine Identität der Objekte vorliegen. Beim Übereinstimmen von variablen Attributbeschreibungen kann jedes beliebige Objekt eingesetzt werden. Wird aber die gleiche Variable bei verschiedenen Konnektoren derselben Transition eingesetzt, müssen die Objekte identisch sein, damit die Transition feuerbar ist.

Eine Marke, die über einen Output-Konnektor fließt, übernimmt die Attribute dieses Konnektors. Werden variable Attribute eingesetzt, die im Input-Konnektor nicht vorkommen, müssen diese in der Funktionsbeschreibung der Transition mit einem Wert versehen werden. Andernfalls wird dem fraglichen Attribut der Wert 'nil' zugewiesen.

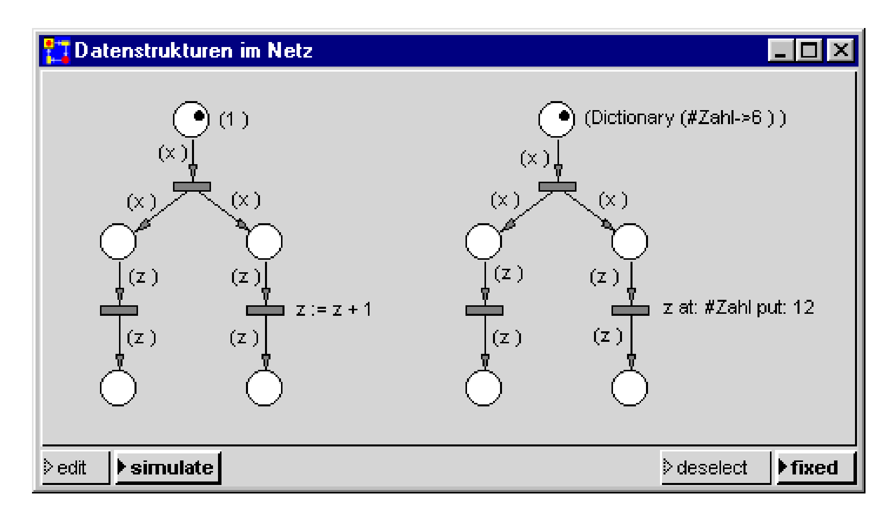


Abb. 3-14:Datenstrukturen im Netz; Ausgangssituation

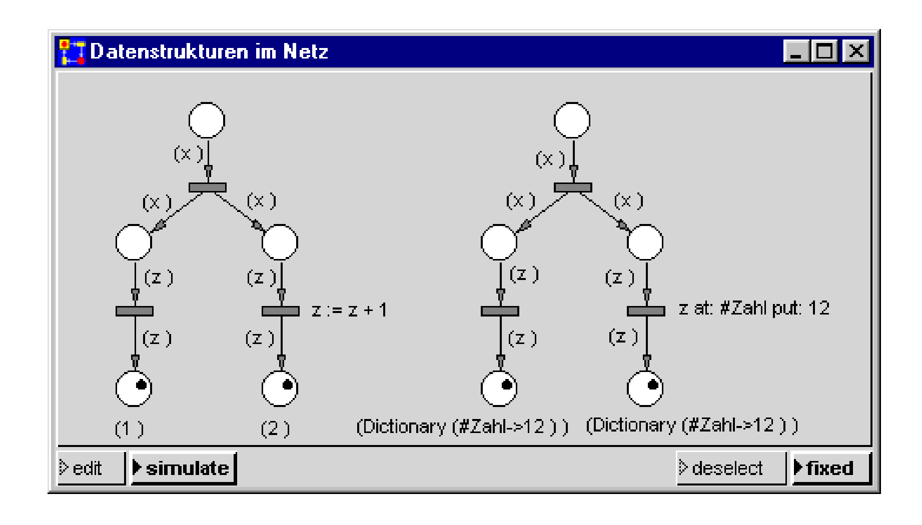


Abb. 3-15: Datenstrukturen im Netz; Endsituation

Wichtiger Hinweis:
Bei Transitionen, bei denen mehrere Ausgangskonnektoren mit dem gleichen Variablennamen versehen sind, ist darauf zu achten, daß es, wenn an der Marke Datenstrukturen (Dictionaries, OrderedCollection, Arrays, Set, etc.) angehängt sind, zu unerwünschten Effekten kommen kann, weil Smalltalk die Referenzen der Variablenamen auf das gleiche Objekt setzt.

Um solche Effekte zu vermeiden, müssen die Objekte mit der Methode 'deepCopy' eindeutig getrennt werden.

Das in Abb. 3-14 bis 3-15 angezeigte Beispiel zeigt eine solche Situation.

Im linken Teilnetz, in dem einfache Daten manipuliert werden, erhält man die erwarteten Ergebnisse.

Das rechte Teilnetz transportiert eine Datenstruktur (Dictionary), der im rechten Zweig ein neuer Wert zugewiesen wird. In beiden

Ergebnisstellen erhält man hier das gleiche geänderte Dictionary (Abb. 3-15).

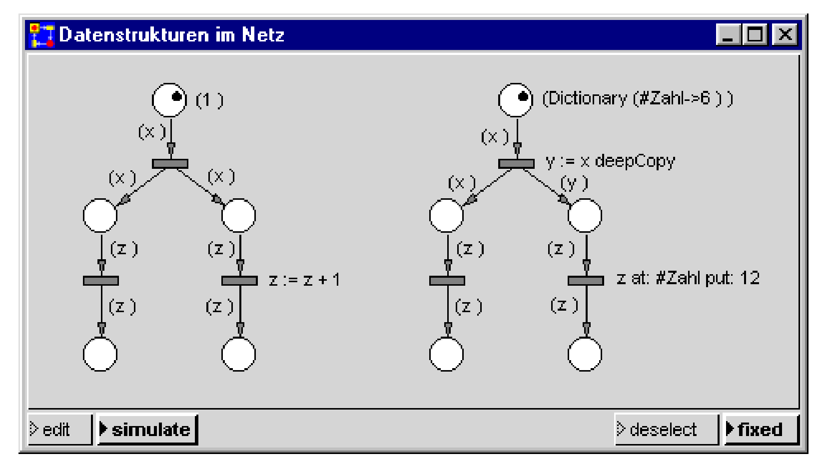


Abb. 3-16: Vermeidung des Konflikts

Um das zu vermeiden, gibt es verschiedene Möglichkeiten. Man kann beispielsweise in der obersten Transition des rechten Netzes mit deepCopy ein zweites Dictionary erzeugen und an den rechten Zweig übergeben. Im linken Zweig wird das Original-Dictionary transportiert (Abb. 3-16).

3.5.4.1 Input-Konnektor-Beispiel

Über den in Abb. 3-17 abgebildeten Input-Konnektor können nur Marken fließen, die drei Attribute ausweisen. Das erste Attribut muß die Integer-Zahl '3' sein. Das zweite Attribut kann einen beliebigen Wert aufweisen. Das dritte Attribut wird über den Bedingungscode eingeschränkt und muß das Symbol '#ok' sein. Die zweite Marke kann nicht über diesen Konnektor fließen.

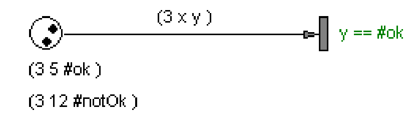


Abb. 3-17:Beispiel für einen attributierten Input-Konnektor

3.5.4.1 Output-Konnektor-Beispiel

Die Marken, die über den in Abbildung 3-18 dargestellten Output-Konnektor fließen, erhalten zwei Attribute. Das erste ist die Integer-Zahl '2', das zweite das Symbol '#no'.

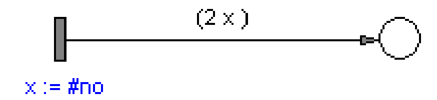


Abb. 3-18: Beispiel für einen attributierten Output-Konnektor

3.5.5 S-Element-Inskriptionen

Stellen und Kanäle können mit Beschriftungen (Kommentare, Namen, usw.) versehen werden. Diese benötigen keine spezielle Syntax und können ganz einfach als Fließtext eingegeben werden. Ihr Einsatz bringt Klarheit in das Netz. Zudem werden sie benötigt, um bei gewissen Botschaften eine Bezugnahme zur Stelle zu erzeugen.



Abb. 3-19: Eine Stellen-Inskription ist ihr Kommentar

3.5.6 Modul-Inskriptionen

Auch die Modul-Inskriptionen sind als Beschriftungen (Kommentare, Namen, usw.) zu verstehen. Diese Beschriftung wird in der Netz-Liste und als Etikette für das Unternetz-Fenster benutzt, zu dem es die Schnittstelle bildet. Hier sollten nur Namen eingesetzt werden, die der Notations-Syntax des unterlegten Betriebssystems entsprechen. Wird einem Modul vom Benutzer kein Name zugewiesen, so

erzeugt PACE automatisch einen Default-Namen (Buchstabe m mit nachfolgender Nummer).

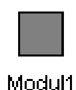


Abb. 3-20:Der Texteintrag eines Moduls ist sein Name

3.5.7 Lange Attribut-Beschreibungen

PACE stellt automatisch zugeordnet zu einer Marke die Markenattribute auf dem Bildschirm dar. Ist eines der Attribute sehr lang, z.B. eine lange Zeichenkette, so kann dies während einer Animation störend wirken. In diesem Fall kann der Anwender selbst eine kürzer Ersatzbeschreibung wählen, die statt der störenden langen Beschreibung als Markenattribut angezeigt wird.

Um die verkürzte Beschreibung zu definieren wird die Klasse „Attribute“ verwendet.

Eine Instanz der Klasse 'Attribute' hat eine sog. Instanz-Variable 'value'. Dieser Variablen kann als Wert das Objekt zugewiesen werden, dessen Beschreibung zu lang ist. Erstellt und gesetzt wird es durch die Botschaft 'value:'. Die Botschaft 'value' (ohne Doppelpunkt!) liefert als Rückgabewert wieder das Objekt. Beim Erstellen eines neuen Attributes kann ein Name angegeben werden, der als Ersatz angezeigt werden soll. Dazu wird die Botschaft 'named:' an die Klasse 'Attribute' gesandt.

Beispiel: Wird bei der Definition eines Markenattributs (im vorliegenden Fall die angegebene Zeichenkette ‘Dieser Text soll ...’) statt der Zeichenkette die folgende Botschaft verwendet:

(Attribute named: 'InfoText') value: ‘Dieser Text soll mit der Marke durch das Netz wandern.'

so wird als Attribut der Marke InfoText angezeigt.

3.6 Zugang zu den Marken einer Stelle

Mit einer Botschaft 'marking' sind die einzelnen Marken einer Stelle erreichbar. Die Stelle sendet als Reaktion auf eine Botschaft, der 'marking' vorausgeht, die dieser Botschaft zugeordnete Information.

3.6.1 Die Botschaft 'tokenNumber'

Als Reaktion auf diese Botschaft wird die Anzahl der sich auf einer Stelle befindenden Marken zurückgesandt.

Wäre in einem Netz beispielsweise eine Stelle mit der Bezeichnung 'eineStelle' vorhanden, so liefern die folgenden Anweisungen als Rückgabewert die Anzahl Marken, die sich auf dieser Stelle befinden:

eineStelle := self placeNamed: ‘eineStelle’.
anzahlDerToken :=eineStelle marking tokenNumber

oder zusammengefaßt:

AnzahlDerToken :=
(self placeNamed: ‘eineStelle’) marking tokenNumber

3.6.2 Die Botschaft 'firstToken'

Rückgabewert ist hier die erste Marke einer Stelle. Wäre in einem Netz beispielsweise eine Stelle mit der Bezeichnung 'eineStelle' vorhanden und befände sich auf ihr mindestens eine Marke mit mindestens einem Attribut, ergäbe 'firstToken' in der folgenden Anweisung als Rückgabewert die erste Marke mit all ihren Attributen in Form eines Arrays.

Diesem können alle Botschaften gesandt werden, die bei einem Array möglich sind. Das erste Attribut der ersten Marke ergibt sich z.B. aus der folgenden Anweisung:

eineStelle marking firstToken first

3.6.3 Die Botschaft 'lastToken'

Diese Botschaft entspricht der Botschaft 'firstToken', nur daß hier die Attribute der Marke einer Stelle geliefert werden, die zuletzt in der Stelle gespeichert wurde. Das letzte Attribut der letzten Marke ergibt sich z.B. aus:

eineStelle marking lastToken last

3.6.4 Die Botschaft 'token:attribute:'

Diese Botschaft liefert das i-te Attribut der k-ten Marke einer Stelle:

eineStelle marking token: k attribute: i

Sind weniger als k Marken vorhanden oder besitzt die k-te Marke kein i-tes Attribut, so wird der Wert nil zurückgegeben.

3.6.5 Die Botschaft 'tokenList'

Als Reaktion auf diese Botschaft wird eine Instanz der Klasse 'OrderedCollection' gesandt, deren Elemente Arrays sind. Diese enthalten der Reihe nach die Attribute der Marken, die sich auf der entsprechenden Stelle befinden. Es sollte allerdings beachtet werden, daß diese Methode bei einer großen Anzahl von Marken auf einer Stelle nicht sehr effizient ist.

Mit der folgenden Anweisung können die Attribute aller Marken einer Stelle gelesen werden:

eineStelle marking tokenList

3.6.6 Hinzufügen von Marken

Normalerweise werden Marken bei der Initialisierung erzeugt. Während des Ablaufs eines Netzes wird ihre Zahl dann mit Netzoperationen vermehrt oder vermindert.

Gelegentlich ist es jedoch auch wünschenswert, Marken über Inskriptionen zu erzeugen. Das ist z.B. der Fall, wenn eine Stelle mit einer sehr großen Anzahl von Initialmarken belegt werden soll oder wenn die Belegung über externe Eingaben (z.B. eine Excel-Tabelle) vorgegeben wird. Neben solchen statischen Vorbelegungsmaßnahmen, die in der Regel im Initialisierungscode durchgeführt werden, gibt es zahlreiche Anwendungen der Methoden während des Ablaufs von Netzen. Beispielsweise kann man bei Eintritt von Ereignissen Marken an verschiedenen Stellen des Netzes erzeugen und damit asynchron (ohne den Aufbau komplexer Netzstrukturen!) auf die Ereignisse reagieren.

Für das Hinzufügen von Marken auf eine Stelle gibt es fünf Methoden, die sich durch die Anzahl der jeweils zu berücksichtigenden Attribute unterscheiden:

addTokenTo:
addTokenTo:with:
addTokenTo:with:with:
addTokenTo:with:with:with:
addTokenTo:attributes:

Als erstes Argument ist jeweils die Stelle, welche die Marke aufnehmen soll, in Form eines Strings einzugeben. Danach sind bei den ersten 4 Methoden 0 bis 3 Argumente anzugeben. Werden mehr als drei Attribute benötigt, so sind die Attribute mit der fünften Methode in Form eines Feldes (Array) nach dem Selektor attributes: anzugeben.

Marken können zu einer Stelle nur hinzugefügt werden, wenn dabei die Kapazität der Stelle nicht überschritten wird. Ob die Marke hinzugefügt werden konnte, kann aus dem Ergebnis der jeweils verwendeten addToken-Methode abgelesen werden. Diese liefert bei erfolgreichem Hinzufügen den Wert: true, andernfalls den Wert: false.

Beispiele:

```
| aPlace |
aPlace := self placeNamed: ‘aPlace’.
self addTokenTo: aPlace.
self addTokenTo: aPlace with: 24.
self addTokenTo: aPlace attributes: #(1 2 3 4 5).
```

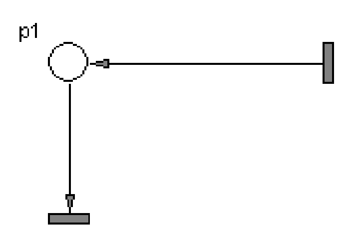
3.6.7 Die Botschaft 'removeAllTokens'

Diese Botschaft löscht alle Marken von einer Stelle.

eineStelle marking removeAllTokens

Das Löschen einer Marke, die zur Vorbereitung des Feuern einer Transition herangezogen wurde (sog. Aktivieren einer Transition), kann einen Laufzeitfehler verursachen.

In dem folgenden Beispiel löscht die Transition am unteren Rand alle Marken, die sich auf der Stelle 'p1' befinden, sobald mindestens drei Marken auf ihr liegen. Dieses wird im Bedingungs-Code der Transition ausgeführt. Hier wird auch gewährleistet, daß die Transition nicht gefeuert wird.




```
m := (self placeNamed: 'p1') marking.
m tokenNumber >= 3 ifTrue: [m removeAllTokens].
false
```

Abb. 3-21: Beispiel für das Entfernen aller Marken einer Stelle

3.6.8 Die Botschaft 'setInitialTokens'

Diese Botschaft setzt die Anfangsbelegung der entsprechenden Stelle. Dazu werden zuerst alle auf der Stelle befindlichen Marken gelöscht.

Warnung: Das Löschen einer Marke, die von einer feuernden Transition benutzt wird, kann einen Laufzeitfehler verursachen!

In dem in Abb. 3-22 dargestellten Beispiel setzt die Transition am unteren Rand die Anfangsbelegung für die Stelle 'p1', sobald mindestens drei Marken auf ihr liegen. Die Abfrage wird im Bedingungs-Code der Transition ausgeführt. Durch das Ergebnis ‘false’ wird gewährleistet, daß die Transition nicht feuert.

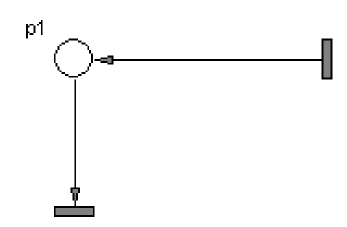


Abb. 3-22: Beispiel für das Setzen einer neuen Initial-Marke, nachdem die Marken einer Stelle zuvor entfernt worden sind.

3.7 Extra-Codes

Jedem System-Modell, das mit PACE spezifiziert wird, kann ein Initialisierungs-, ein Fortsetzungs-, ein Unterbrechungs- und ein Beendigungs-Code zugewiesen werden. Alle vier Code-Segmente sind Smalltalk-Codes. Sie können beispielsweise eingesetzt werden, um globale Variablen zu besetzen, um Files zu öffnen oder zu schließen, usw.

- Der Initialisierungscode wird durch den Menüpunkt 'initialize' (Menü der mittleren Maustaste in einem Netzfenster, das sich im Simulationsmode befindet) ausgeführt. Der Initialisierungscode wird ausgeführt, bevor irgendeine Transition feuert.

- Der Fortsetzungscode wird ausgeführt, wenn ein Simulationslauf nach einer Unterbrechung fortgesetzt wird.

- Der Unterbrechungscode wird ausgeführt, wenn ein Simulationslauf unterbrochen wird.

- Der Beendigungs-Code wird ausgeführt, wenn:

  - keine Transition mehr feuern kann (es erscheint die System-Meldung "End of simulation")

  - die Anweisung 'self terminate' oder ‘self restart’ in einem Transitions-Code ausgeführt wurde

  - manuell der Menüpunkt 'terminate' im Simulator-Hauptmenü ausgelöst wurde.

Für das Einsetzen von Extra-Codes ist im Net-Editor-Menü der PACE-Hauptleiste der Menüpunkt 'extra codes' vorgesehen.

3.7.1 Zugang zu einer Stelle in einem Extra-Code

Mit der Botschaft

self placeNamed: 'Name einer Stelle'

kann eine bestimmte Stelle während der Ausführung eines Extra-Code identifiziert werden. Auf diese Stelle sind alle Methoden anwendbar, die in diesem Kontext sinnvoll sind.

Die Identifizierung von Stellen während der Initialisierung eines Modells ist bei großen Netzen angezeigt, wenn entfernt liegende Stelle häufig angesprochen werden. Die Vorab-Identifizierung von Stellen vermeidet lange Suchläufe während der Modellausführung und führt deshalb zu erheblich schneller ablaufenden Modellen.

Beispiel: Vorbesetzen der Kapazität einer Stelle:

(self placeNamed: 'stelle1') setCapacity: 5.

3.7.2 isRestarted-Abfrage

Bei mehrfachem programm-gesteuertem Durchlauf eines Modells sollen häufig bestimmte Botschaften (etwa zur Initialisierung von Variablen) nur beim ersten Durchlauf verwendet werden.

Mit der Botschaft:

self isRestarted

kann abgefragt werden, ob das Modell erneut gestartet wurde (siehe auch Abschnitt 3.5.3 und Abb. 3-12).

3.8 PACE-Modell-Variablen

Die Variablen, die in einem PACE-Modell benutzt werden, enthalten immer ein Objekt. Für alle Variablen mit Ausnahme von Modul-Variablen kann mit dem Zuweisungszeichen := einer Variablen ein Objekt zugewiesen werden. Modul-Variablen werden immer über Botschaften angesprochen.

Bei der Vereinbarung einer Variablen weist PACE ihr automatisch ein Objekt der Klasse 'nil' zu. Da Smalltalk als Programmiersprache kein Datentypkonzept kennt, kann einer Variablen jedes Objekt zugewiesen werden.

Jede Variable muß, bevor sie verwendet werden kann, deklariert werden. Wird in einem inskribierten Smalltalk-Code eine nicht deklarierte Variable verwendet, so erscheint eine Meldung, die den Anwender auffordert, die Variable interaktiv zu deklarieren.

Die verschiedenen Variablentypen werden nachfolgend beschrieben.

3.8.1 Konnektor-Variablen

Konnektor-Variablen treten als Attribute einer Konnektor-Inskription auf. Je nach Konnektor-Typ werden diese Variablen Input- oder Output-Variablen genannt. Sie sind ein Attribut der Konnektor-Inskription. Konnektor-Variablen benötigen innerhalb der mit dem jeweiligen Konnektor verbundenen Transition keine Deklaration. Diese Variablen sind lokal, das heißt innerhalb des Transitionscodes und den mit der Transition verbundenen Konnektoren bekannt. Jedes Ausführen einer Transitions-Inskription beginnt mit der Instantiierung aller Input-Variablen.

Das dafür benötigte Objekt wird von der Input-Marke geliefert. Ohne genaues Spezifizieren innerhalb der Transitions-Inskription werden die Output-Variablen defaultmäßig mit dem Objekt 'nil' belegt. Beim

Einsatz einer Input-Variablen auf einem Output-Konnektor wird das Objekt, mit dem sie instantiiert wurde, ein Attribut der Output-Marke, falls der Variablen durch die Transitions-Inskription kein neues Objekt zugewiesen wurde. Vom Objekt selbst wird keine Kopie erstellt. Das in der Variablen enthaltene Objekt kann, falls die benutzte Klasse dies erlaubt, durch das Senden einer bestimmten Botschaft verändert werden.

Die folgende Abbildung zeigt einen Input- und einen Output-Konnektor. Im ersten sind zwei Input-Variablen enthalten (x, y), von denen eine (y) im Code des Output-Konnektoren zusammen mit der eigentlichen Output-Variablen (z) wieder eingesetzt wird.

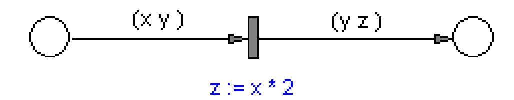


Abb. 3-23: Konnektor-Variablen

3.8.2 Temporäre Variablen

In PACE können temporäre Variablen innerhalb einer Transitions-Inskription, der Definition der Attribute einer Initial-Marke und der Extra-Codes vorkommen. Als Teil von Transitions-Inskriptionen werden sie automatisch deklariert, nachdem der Anwender dies interaktiv bestätigt hat.

Überall sonst müssen temporäre Variablen zwischen senkrechten Strichen (vertical bar '|') deklariert werden. Werden mehrere temporäre Variablen in einer Deklaration vereinbart, so sind sie durch Leerzeichen voneinander zu trennen. Der Name von temporären Variablen muß mit einem Kleinbuchstaben beginnen.

Temporäre Variablen von Transitionen weisen ein ganz spezielles Verhalten auf: Sie bleiben über eine gewissen Zeit hinaus bestehen, nämlich über die Dauer der Transitions-Zeitverzögerung. Wird einer temporären Variablen z.B. innerhalb des Delay-Codes einer Transition ein Wert zugewiesen, so ist dieser Wert beim Ausführen

des Aktions-Codes nach Ablauf der Zeitverzögerung immer noch gültig.

3.8.3 Modul-Variablen

Das Modul-Variablen-Konzept ist in Zusammenhang mit hierarchischen Petri-Netzen sehr hilfreich, insbesondere dann, wenn Module einzeln abgespeichert und in einem anderem Kontext wiederverwendet werden. Modul-Variablen verhalten sich aber innerhalb eines Inskriptionscodes nicht wie gewöhnliche Variablen, das heißt, sie werden nicht durch Identifikatoren dargestellt, sondern sind nur über Botschaften zugänglich.

Jeder Modul-Variablen ist ein Schlüsselwort zugeordnet, das ein Smalltalk-Symbol ist. Sie kann mit einem Initialisierungs-Code versehen werden. Während ihrer Ausführung kann eine Modul-Variable gelesen, geschrieben, initialisiert und über Änderungen der in ihr gespeicherten Objekte informiert werden.

3.8.3.1 Zugang zu einer Modul-Variablen

Die folgende Botschaft hat eine Modul-Variable als Rückgabewert:

self at: #varName

Sie kann innerhalb des Transitions-Inskriptionscodes und in Extra-Codes eingesetzt werden.

Bei einem Zugriff auf die Variable mit der at:-Anweisung irgendwo innerhalb der Modulhierarchie wird das den Zuweisungscode enthaltende Modul nach dem entsprechenden Schlüsselwort abgesucht. Verfügt das Modul über eine Variable mit dem gesuchten Schlüsselwort, wird auf diese Variable zugegriffen. Ist dies nicht der Fall, so wird das hierarchisch nächst höhere Modul abgesucht. Die Suche wird auf diese Weise bis zur Wurzel fortgesetzt. Falls beim Suchpfad bis zur Wurzel keine Variable gefunden wurde, tritt ein Laufzeitfehler auf.

Befindet sich die at:-Botschaft in einem Extra-Code, so wird die Modulvariable im Hauptmodul erwartet.

Von den Extra-Codes und von Transitionscodes aus kann selektiv auf Modulvariablen in beliebigen Modulen mit der Methode:

self moduleNamed: aSelectionString at: #varName

zugegriffen werden. Darin ist aSelectionString ein gültiger Pfad vom Hauptmodul zu dem Modul, in dem die anzusprechende Variable #varName vereinbart sein muß. In dem String aSelectionString sind die Module absteigend vom Hauptmodul aus nacheinander durch Punkte getrennt anzugeben.

Beispiel: 'root.mod1.mod2'

Es sollte beachtet werden, daß der Rückgabewert der Botschaften "at:" und "moduleNamed:at:" ein die Modul-Variable darstellendes Objekt ist und nicht ihr aktueller Wert. Eine Modul-Variable speichert eine Anzahl weiterer Objekte, wovon eines ihr Wert ist.

3.8.3.2 Lesen von Modul-Variablen

Die **value**-Botschaft hat den Wert der Modul-Variablen als Rückgabewert:

Beispiele:

```
(self at: #anzahl) value.
(self moduleNamed: 'rootMod.mod2' at: #anzahl) value.
```

3.8.3.3 Schreiben von Modul-Variablen

Mit der **value:**-Botschaft kann einer Modul-Variablen ein Wert (anObject) zugewiesen werden:

Beispiele:

```
(self at: #anzahl) value: anObject.
(self moduleNamed: 'rootMod.mod2' at: #anzahl)
```

value: anObject.

Die Smalltalk-Botschaft ‘changed’ (siehe Abschnitt 3.8.3.5 weiter unten) wird nach der Zuweisung automatisch an die Variable gesandt, damit offene Fenster jeweils auf den neuesten Stand gebracht werden.

3.8.3.4 Initialisieren von Modul-Variablen

Die **initialize**-Botschaft wertet den Initialisierungscode der Modul-Variablen aus und weist ihr das Ergebnis zu:

Beispiele:

```
(self at: #anzahl) initialize.
(self moduleNamed: 'rootMod.mod2' at: #anzahl) initialize.
```

3.8.3.5 Aktualisieren der Anzeige

Die **change**-Botschaft bringt alle Fenster, die den aktuellen Wert der Modul-Variablen anzeigen, auf den neuesten Stand:

Beispiele:

```
(self at: #anzahl) changed.
(self moduleNamed: 'rootMod.mod2' at: #anzahl) changed.
```

3.8.4 Globale Variablen

Vor dem unkritischen Einsatz globaler Variablen ist abzuraten. Anstelle globaler Variablen können häufig Modul-Variablen verwendet werden.

Innerhalb von PACE können globale Variablen überall eingesetzt werden. Diese werden von PACE auch automatisch deklariert, wenn ein Modell, das globale Variablen enthält, als .net-File gespeichert und danach wieder in PACE geladen wird. Dabei gehen die den globalen Variablen zugewiesenen Werte aber verloren und müssen deshalb vor der Verwendung einer globalen Variablen im Inskriptionscode zugewiesen werden.

Die Namen der globalen Variablen müssen mit einem Großbuchstaben beginnen. Variablen-Namen dürfen nicht mit Smalltalk-Klassen-Namen übereinstimmen. Durch Ausführen der folgenden Anweisung in einem Workspace kann abgeklärt werden, ob der Name bereits verwendet wurde (Rückgabewert 'true') oder nicht (Rückgabewert 'false')

Smalltalk includesKey: #NameDerGlobalenVariablen.

Globale Variablen können nicht als Inskriptionen für Konnektoren benutzt werden. Beim erstmaligen Benutzen einer globalen Variablen, muß diese interaktiv mit Hilfe eines Menüs als 'global' deklariert werden. Dieses Menü wird beim Erkennen nicht vereinbarter Variablen während der Übersetzung automatisch angezeigt.

Globale Variablen sollten mit Vorsicht verwendet werden, wenn von ihnen Feuerungsbedingungen abhängen.

3.9 Feuern einer Transition

In PACE kann eine Transition gefeuert werden, sobald diese dazu aktiviert wird. Das Feuern muß allerdings nicht unmittelbar nach dem Aktivieren der Transition erfolgen, weil manchmal mehrere Transitionen gleichzeitig feuerbereit sein können. In einem solchen Fall wird der Simulator, falls nicht in einer Inskription eine bestimmte Reihenfolge vorgegeben wurde, bei der Feuerungs-Reihenfolge willkürlich vorgehen. Alle gleichzeitig feuerbereiten Transitionen werden aber gefeuert, bevor die Simulationszeit weitergezählt wird.

Beim Feuern einer Transition werden die Marken der Input-Stellen, die das Feuern initialisiert haben, von diesen abgezogen. Die Marken werden dann durch Ausführen der Transitions-Inskription weiterverarbeitet. Als Resultat wird schließlich eine oder werden mehrere neue Marken erzeugt und in den Output-Stellen gespeichert. Die Attribute dieser neuen Output-Marken werden dabei von den Inskriptionen der Output-Konnektoren bestimmt.

Damit eine Transition für eine Feuerung aktiviert werden kann, müssen die Bedingungen erfüllt sein, die im folgenden beschrieben sind.

3.9.1 Bedingung 1: Marken auf allen Input-Stellen

Auf jeder Input-Stelle, die nicht durch einen Inhibitor mit einer Transition verbunden ist, muß mindestens eine Marke liegen, deren Attribute mit den Spezifikationen in der Konnektor-Inskription verträglich sind.

3.9.2 Bedingung 2: Inhibitoren

Die Inskriptionen der Marken, die sich auf einer durch einen Inhibitor mit einer Transition verbundenen Inputstelle befinden, dürfen nicht mit denen des Inhibitors übereinstimmen, oder die Inputstelle muß leer sein.

3.9.3 Bedingung 3: Bedingungs-Code

Bei der Abarbeitung und bei der Instantiierung der Input-Variablen dürfen die zugewiesenen Werte im Bedingungs-Code der Transitions-Inskription nicht 'false' als Rückgabewert liefern.

3.9.4 Bedingung 4: Zeitverzögerungen

Die angegebene Zeitverzögerung der Transition muß bereits verstrichen sein. Dies bedeutet, daß die Aktivierung mindestens so viele Zeiteinheiten abwarten muß, wie die Zeitverzögerung aufweist.

3.9.5 Bedingung 5: Kapazität der Output-Stellen

Die Transition-Inskriptionen sollten so beschaffen sein, daß nach ihrer Abarbeitung die Kapazitätsbeschränkung der Output-Stellen nicht überschritten wird.

3.9.6 Übereinstimmung von Marken und Konnektor-Inskriptionen

Eine Marke stimmt mit den Konnektor-Inskriptionen überein, wenn alle ihre Attribute mit denen der Konnektor-Inskriptionen übereinstimmen. Innerhalb einer Konnektor-Inskription können Variablen auf irgend ein Objekt zeigen. Werden aber die gleichen Variablen in den Inskriptionen verschiedener Konnektoren der gleichen Transition eingesetzt, müssen diese Variablen immer auf das gleiche (identische) Objekt verweisen. Konstante Konnektor- Inskriptionen müssen mit dem von der Inskription selbst erzeugten Objekt übereinstimmen.

Es sollte vermieden werden, die Aktivierung einer Transition von globalen Variablen abhängig zu machen. Wird eine solche Transition

nach einem Test von PACE als nicht feuerbar taxiert, wird diese vom Simulator erst wieder getestet, wenn sich die Belegung innerhalb der Umgebung der betroffenen Transition geändert hat.

3.10 Zeitmodellierung

PACE erlaubt die Modellierung des Zeitverhaltens eines System-Modells. Dies geschieht, indem das Feuern einer Transition verzögert wird. Im folgenden Abschnitt werden die Einzelheiten beschrieben, die bei der Zeitmodellierung zu berücksichtigen sind.

3.10.1 Zeitverzögerungs-Syntax

Eine Zeitverzögerung wird durch den Delay-Code einer Transition bestimmt.

Die in der folgenden Abbildung dargestellte Transition kann erst gefeuert werden, wenn die Marke mit dem Attribut 2 auf der Stelle erschienen ist, und darauf 4 Simulations-Zeiteinheiten vergangen sind. Der Wert dieser Zeitverzögerung kann auch ein Wert sein, der vorher innerhalb des Delay-Codes errechnet wird. Er kann auch von den Attributen der Input-Marken abhängen.

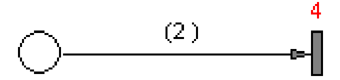


Abb. 3-24:Transitions-Inskription mit einer Zeitverzögerung

3.10.2 Mehrfach-Aktivierungen

Es kann vorkommen, daß mehrere Marken gleichzeitig auf das Ablaufen einer Zeitverzögerung für dieselbe Transition warten. Weiter kann eine Marke auf das Ablaufen der Zeitverzögerung von gleichzeitig mehreren Transitionen warten. In der folgenden Abbildung warten beide Marken gleichzeitig auf das Ablaufen der Zeitverzögerung.

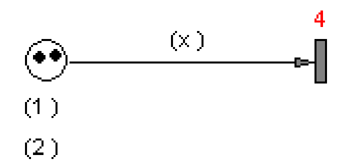


Abb. 3-25: Zwei Marken der gleichen Stelle warten auf das Ablaufen der Zeitverzögerung

3.10.3 Kapazitätsbeschränkung für Output-Stellen

Eine Kapazitätsbeschränkung für Output-Stellen kann wie eine Zeitverzögerung zu einem zeitlichen Hinausschieben des Feuerns einer Transition führen. Auch wenn eine Zeitverzögerung abgelaufen ist, muß eine Aktivierung weiter warten, wenn wegen einer Kapazitätsbeschränkung das Feuern der Transition weiterhin nicht möglich ist. Sobald eine Marke von der blockierenden Output-Stelle abgezogen wird, kann die verzögerte Transition gefeuert werden.

Kapazitätsbeschränkungen verursachen ein ähnlich verzögerndes Verhalten wie eine Zeitverzögerung.

Eine Kapazitätsbeschränkung einer Stelle kann während der Editierung einer Stelle per Menü, aber auch über den Transitionscode festgelegt werden.

3.10.3.1 Die Botschaft 'getCapacity'

Diese Botschaft liefert den aktuellen Wert der Kapazität einer Stelle:

eineStelle getCapacity

3.10.3.2 Die Botschaft 'setCapacity:'

Diese Botschaft ändert den aktuellen Wert der Kapazität einer Stelle auf den im Argument angegebenen positiven Ganzzahlwert:

eineStelle setCapacity: anInteger

Das Argument 0 bedeutet, daß die Stelle beliebig viele Marken aufnehmen darf.

Die Botschaft darf nur verwendet werden, wenn zum Zeitpunkt der Kapazitätsänderung keine Marken in der Stelle gespeichert sind.

3.10.4 Inhibitoren-Semantik

Bei der Verwendung von zeitverzögerten Transitionen, zu denen Inhibitoren als Input-Konnektoren führen, ist Vorsicht angebracht. Das Feuern einer solchen Transition ist für einen späteren Zeitpunkt vorgesehen, wenn kein Inhibitor die Aktivierung verhindert. Kommt nun aber eine Marke zu einem Zeitpunkt an eine inhibierte Input-Stelle, zu dem die Transition bereits auf das Ablaufen der Zeitverzögerung wartet, so wird das Feuern vom Inhibitor nicht verhindert.

Die Implementation dieser Lösung wurde vor allem aus Effizienz-Gründen gewählt. Sie entspricht der Semantik von Inhibitoren die zu Transitionen führen, welche Output-Stellen mit Kapazitäts-beschränkungen aufweisen.

3.10.5 Marken-Rücksetzungen

Eine Marke, die eine Zeitverzögerung abwartet, ist für die verzögernde Transition nicht reserviert. Es kann vorkommen, daß diese Marke von anderen, nicht verzögerten Transitionen von ihrer Stelle abgezogen und verbraucht wird. Da die zu verarbeitende Marke dann beim Ablaufen der Verzögerungszeit nicht mehr vorhanden ist, kann die verzögerte Transition nicht mehr gefeuert werden.

3.10.6 Allgemeine Regeln

Folgende Regeln sollten normalerweise beachtet werden:

1. Eine verzögerte Transition soll höchstens einen Input-Konnektor und keinen Inhibitor besitzen!

2. Eine Input-Stelle und eine verzögerte Transition sollen nicht über einen doppelten Konnektor verbunden werden!

Um unliebsame Überraschungen zu vermeiden, sollte auf die Einhaltung diese beiden Regeln nur in Sonderfällen verzichtet werden.

Eine verzögerte Transition ohne Input-Konnektor kann manchmal eingesetzt werden, um Ereignisse zu einem bestimmten Zeitpunkt zu erzeugen, beispielsweise um eine Uhr zu modellieren.

Von der ersten Regel kann eine Ausnahme gemacht werden, wenn in einem Netz eine Transition durch Doppelkonnektor mit einer zusätzlichen Kontroll-Stelle versehen ist, und dabei ausdrücklich nur eine aktivierende Marke als Input gewünscht wird. Das Beispiel in Abbildung 3-26 zeigt eine solche Situation.

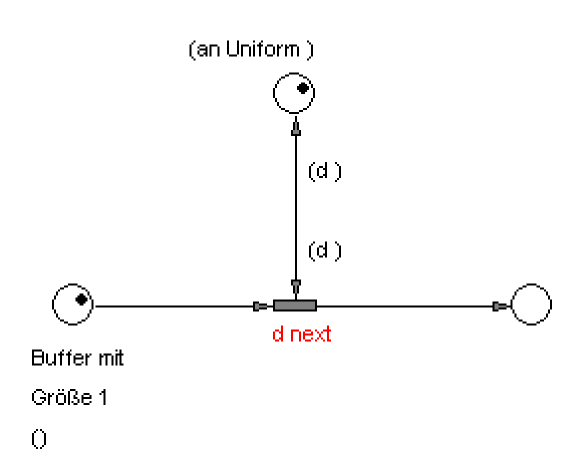


Abb. 3-26: Eine Ausnahme von den genannten Regeln

3.10.7 Durch Zufall beeinflusste Zeitverzögerungen

Mit PACE kann das stochastische Verhalten eines Systems durch den Einsatz einer entsprechenden Smalltalk-Klasse modelliert werden (siehe Kapitel 10). Die Realisierung besteht zum Beispiel darin, daß eine Instanz einer Zufallsverteilung dieser Klasse ein Marken-Attribut darstellt. Wird dann einer solchen Instanz die Botschaft 'next' gesandt, gibt diese als Reaktion auf 'next' den nächsten Wert der Verteilung zurück. Die Marke mit der Verteilung als Attribut wird immer wieder auf die gleiche Stelle zurückfließen (siehe dazu die Beispiele in diesem Kapitel).

Stochastische Zeitverzögerungen könnten auch durch Speichern der Verteilung in einer globalen Variablen gelöst werden.

3.10.8 Zugang zur aktuellen Simulations-Zeit

Die aktuelle Simulations-Zeit ist über die globale Variable

CurrentTime

zugänglich. Diese globale Variable kann in jedem Smalltalk-Code verwendet werden. Ein Beispiel dazu zeigt Abb. 3-29.

3.10.9 Beispiele mit zeitabhängigen Transitionen

3.10.9.1 Zeitsperre (Timeout)

In Abbildung 3-27 wird ein Beispiel einer Zeitsperre gezeigt. Gewartet wird auf eine Meldung (Marke) in der Stelle rechts unten innerhalb von 10 Zeiteinheiten.

Bei rechtzeitiger Lieferung dieser Meldung wird eine Marke auf die Stelle 'Übertragung OK' gelegt, um die hierarchisch höhere Ebene über die erfolgreiche Übertragung zu informieren.

Sollte aber innerhalb von 10 Zeiteinheiten keine Meldung eintreffen, so wird die hierarchisch höhere Ebene über die negative Übertragung informiert (Stelle 'Übertragung nicht erfolgt'). Die untere Transition mit dem Inhibitor als Input-Konnektor verbraucht die Meldungs-Marken, die nach Ablauf der Zeitsperre auf die Stelle rechts unten gelangen.

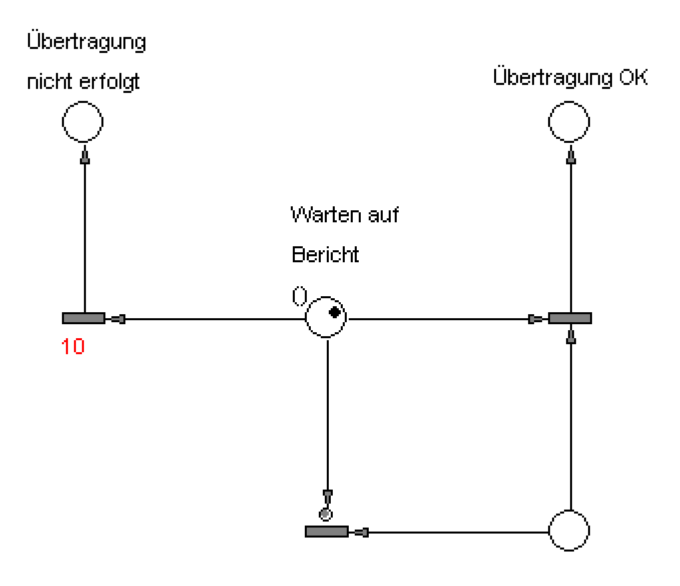


Abb. 3-27:Beispiel für die Modellierung von Zeitsperren

3.10.9.2 Gleichverteilte Feuerungs-Intervalle

In Abbildung 3-28 wird gezeigt, wie eine Gleichverteilung der Feuerungs-Intervalle einer Transition spezifiziert werden kann.

Die Marke der Stelle links oben hat als Attribut eine Instanz der Klasse 'Uniform' (siehe Kapitel 10). Beim Feuern der Transition wird der Konnektor-Variablen diese Instanz zugewiesen. Als Reaktion auf die empfangene Botschaft 'next' liefert sie gleichverteilte Werte im Intervall von 10 bis 20. Die Zuweisung des nächsten Wertes erfolgt

im Zeitverzögerungs-Code innerhalb der Transition. Danach wird die veränderte Instanz der Gleichverteilung wird wieder in die Stelle links oben zurücktransportiert und der Vorgang wiederholt sich.

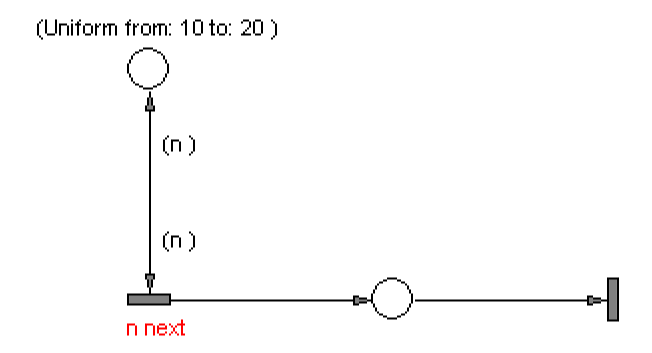


Abb. 3-28: Beispiel für normal verteilte Feuerungs-Intervalle

3.10.9.3 Zeitmarken

In diesem Beispiel (Abb. 3-29) setzt die Transition, die sich im unteren Bereich befindet, eine Zeitmarke auf die Output-Stelle.

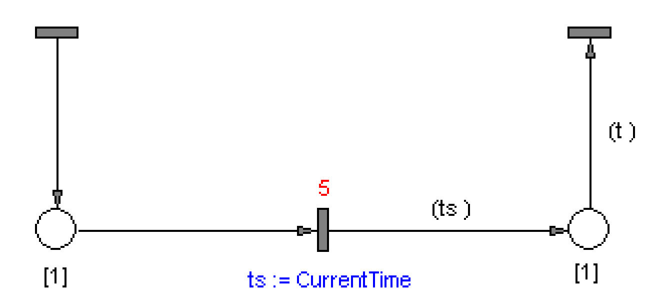


Abb. 3-29: Eine Transition setzt eine Zeitmarke auf ihre Output-Stelle

4 BENUTZERSCHNITTSTELLE

Die Benutzerschnittstelle von PACE ist leicht zu erlernen und orientiert sich weitgehend an MS-Windows. Sie ermöglicht eine effiziente Bedienung von PACE und ist auf allen Plattformen, auf denen PACE verfügbar ist, nahezu identisch.

4.1 Prinzipielle Arbeitsweise

Eine PACE-Funktion auszuführen, beinhaltet im wesentlichen folgende Schritte:

1. Fenster aktivieren, in dem die Funktion zu finden ist, die ausgeführt werden soll (falls dieses noch nicht aktiviert ist).

2. Elemente auswählen, auf welche die Funktion angewandt werden soll (falls dies gewünscht wird).

3. Drücken der Maustaste (im Normalfall die rechte), um ein Menü erscheinen zu lassen, das die zur Verfügung stehenden PACE-Menüfunktionen beinhaltet.

4. Auswählen der auszuführenden Funktion.

4.2 Maus

Die PACE-Benutzerschnittstelle wird normalerweise mit einer Drei-Tasten-Maus bedient. Bei den Systemen, die mit einer Maus arbeiten, die weniger als drei Tasten hat, kann die fehlende Taste (die fehlenden Tasten) zusammen mit der Tastatur simuliert werden. PACE versucht jedenfalls so viele wie möglich von den auf der Hardware verfügbaren und auch wirklich ausführbaren Funktionen durch Maustasten zur Verfügung zu stellen.

Die linke Maustaste
Die linke Maustaste wird eingesetzt, um Text oder graphische Elemente in einem Netz zu selektieren.

Bei Listen mit Textzeilen wird die Selektion normalerweise durch Drücken einer Bestätigungstaste abgeschlossen. Sie kann aber auch durch Doppelklick auf die zu selektierende Zeile beendet werden.

Die mittlere Maustaste
Die mittlere Maustaste, die bei den heute gängigen Mäusen durch die Druckfunktion des Mausrads bereitgestellt wird öffnet das Pop-up-Menü mit den Fensterfunktionen an jeder beliebigen Stelle im Fenster.

Die rechte Maustaste
Durch Drücken der rechten Maustaste wird das dem Kontext entsprechende Pop-up-Menü für PACE-Aktivitäten geöffnet.

4.3 Fenster

Die PACE-Benutzerschnittstelle ist fensterorientiert. Meist wird mit mehreren Fenstern gearbeitet. Jedes Fenster zeigt einen anderen Aspekt des Systems.

Im Prinzip können beliebig viele Fenster gleichzeitig geöffnet werden. Die Anzahl der gleichzeitig geöffneten Fenster wird lediglich durch das Betriebssystem begrenzt. Zu einigen Fenstern, zum Beispiel zu Text-Fenstern und zu Dialog-Boxen, können verschiedene untergeordnete Fenster geöffnet werden.

4.3.1 Fenster-Funktionen

Die Fenster-Funktionen werden mit der mittleren Maustaste angezeigt, wenn sich der Cursor innerhalb eines Fensters befindet.

new label
Öffnet ein Dialog-Fenster. Darin kann das Fenster mit einem neuen Namen versehen werden.

refresh
Der Inhalt des aktiven Fensters wird neu aufgebaut.

move
Verschiebt das angezeigte Fenster auf dem Bildschirm, ohne seine Größe zu verändern.

resize
Gibt die Möglichkeit, die Fenster-Größe zu verändern.

front
Das angezeigte Fenster wird vor allen anderen Fenster im Vordergrund platziert.

back
Das angezeigte Fenster wird hinter allen anderen Fenstern platziert.

close

Schließt das Fenster. Diese Funktion sollte immer dann benutzt werden, wenn ein Fenster nicht mehr gebraucht wird. Das Fenster kann ja, falls erforderlich, zu einem späteren Zeitpunkt wieder geöffnet werden.

In den meisten Fällen erfragt PACE vor dem Schließen des Fensters per Dialog, ob das zu schließende Fenster gesichert werden soll oder sichert die Änderungen in einem Zwischenpuffer. Damit wird sichergestellt, daß keine im Fenster gespeicherte Informationen verloren gehen. Es wird empfohlen, nicht zu viele Fenster gleichzeitig offen zu halten, da sonst leicht die Übersicht verloren gehen kann und sich das Fenster-Management verlangsamt. Werden in einem Modell viele Fenster benötigt, so ist eine Aufteilung in Szenen zweckmäßig. Szenen ermöglichen das Anzeigen und Verbergen von Fenstersätzen jeweils mit einem einzigen Mausklick.

4.3.2 Fenster-Typen

Die folgende Auflistung beschreibt die wichtigsten Fenster-Typen von PACE.

4.3.2.1 Netz-Fenster

Ein Netz-Fenster zeigt die graphische Darstellung eines Netzes. In diesem Fenster können Netze modelliert und animiert werden. Jedes Fenster kann an sein Netz gebunden werden ('fixed'). Gebundene Fenster zeigen immer das gleiche Netz. In der Regel sind mehrere dieser Fenster-Typen gleichzeitig offen. Der Simulator steuert die Animation automatisch von einem gebunden Fenster zu einem anderen.

Daneben gibt es nicht gebundene Netz-Fenster. Von diesem Fenster-Typ darf nur ein einziges Fenster zu einem Zeitpunkt offen sein ('non fixed'). Der Simulator zeigt darin das Netz an, in dem gerade eine Transition gefeuert wird. So kann das dynamische Verhalten des gesamten Modelles beobachtet werden. Das Netz, welches als erstes im nicht gebundenen Netz-Fenster angezeigt werden soll, kann entweder in der Netz-Liste (wenn bei den Optionen im ‘view’-Menü der Developer-Leiste nicht 'fixed' als

Default-Modus beim Fensteröffnen gesetzt wurde), oder durch den entsprechenden Aktivierungsbefehl (z.B. 'supernet' oder 'subnet') ausgewählt werden.

Ein Netz-Fenster besteht aus folgenden Teilen:

- **Das eigentliche Netz-Fenster:** Hier wird das Netz modelliert, angezeigt und animiert.
- **'edit'-Schalter**: Schaltet in den Editier-Modus.
- **'simulate'-Schalter**: Schaltet in den Simulations-Modus.
- **'deselect'-Schalter**: Setzt alle Selektionen im Netzfenster zurück.
- **'fixed'-Schalter**: Erlaubt ein Fenster als zum Netz gebunden ('fixed') oder nicht gebunden ('non fixed') zu definieren.

Der Hintergrund der Netz-Fenster wird normalerweise mit einer vom Benutzer festlegbaren Farbe dargestellt. Um die Modelldarstellung interessanter zu gestalten und um die Darstellung einer aktuellen Umgebung graphisch zuzuordnen, kann stattdessen auch ein Hintergrundbild verwendet werden (siehe Kapitel 7).

4.3.2.2 Marken-Fenster

Marken-Fenster werden benutzt, um die Marken einer Stelle neu zu setzen und zu initialisieren, oder um deren aktuellen Werte zu verändern. Die folgende Abbildung zeigt die Initialisierungsbelegung einer Stelle mit 10 Marken. Die dritte Marke ist selektiert und hat 2 Attribute. Das zweite dieser Attribute ist das Symbol #maschine2. Dem Token wurde eine Anfangsikone zugeordnet, die dem Token von der Stelle ‘Teile-Eingang’ bis zur ersten Transition verwendet wird.

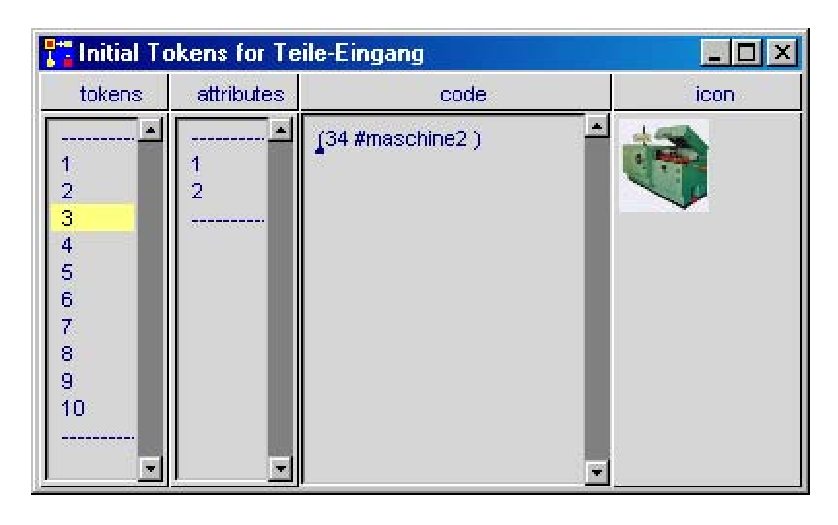


Abb. 4-1: Marken-Fenster

4.3.2.3 Zeit-Diagramm-Fenster

Ein Zeit-Diagramm-Fenster zeigt ein Balken- oder Liniendiagramm, in dem ein vom Benutzer definierter Aspekt einer Stelle oder eines T-Konnektors in Abhängigkeit von der Simulationszeit dargestellt wird.

4.3.2.4 Normale Diagramm-Fenster

In einem normalen Diagramm-Fenster werden Abhängigkeiten zwischen Programmgrößen, die bei der Simulation berechnet werden, z.B. als Balken- oder Tortendiagramm dargestellt (siehe Kap. 6).

4.3.2.5 Graphische Ein/Ausgabe-Fenster

Für die Ein/Ausgabe von Daten gibt es in PACE zahlreiche graphische Ein/Ausgabe-Fenster. Diese können sowohl für die Dateneingabe als auch für die Visualisierung von Simulationsergebnissen

verwendet werden (siehe Kap. 6).

4.3.2.6 Text-Fenster

In den Text-Fenstern werden die Textteile eines PACE-Modelles angezeigt oder verändert. Das Menü mit den Funktionen des Text-Editors steht durch Drücken der rechten Maustaste zur Verfügung. In Abbildung 4.2 ist das Menü mit den Text-Editor-Befehlen zu sehen. Es wird gerade ein Kommentar für ein S-Element angezeigt.

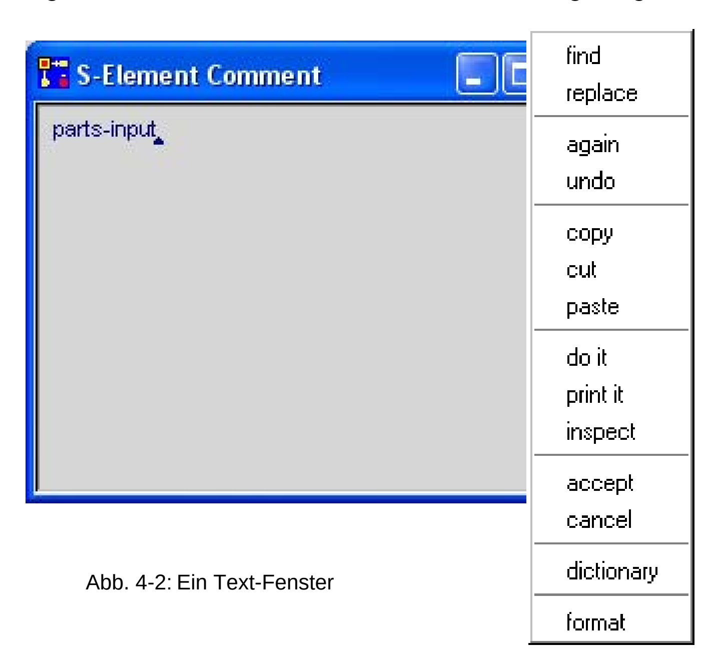


Abb. 4-2: Ein Text-Fenster

4.3.2.7 Listen-Fenster

Listen-Fenster zeigen Listen der Marken, Attribute, Manualeinträge, Netze usw. an. Mit der rechtenn Maustaste lassen sich die Listen-Befehle aktivieren und mit der linken Maustaste können die einzelnen Einträge der Listen an- beziehungsweise abgewählt werden. Abbildung 4.3 zeigt ein Listen-Fenster, welches die Netze eines PACE-Modells auflistet.



Abb. 4-3: Ein Netz-Listen-Fenster

4.3.2.8 Optionen-Fenster

Diese Fenster erlauben die Spezifikation von Parametern für PACE insgesamt oder für einzelne Module. Jeder Parameter wird in einem eigenen Unterfenster definiert. Werden boolesche Werte verlangt, sind im Unterfenster zwei Schalter enthalten, die erlauben, den Wert zu setzten oder zurückzusetzen.

Ein Schalter wird durch Anklicken der linken Maustaste aktiviert und wird dann erhaben dargestellt. Wird die Eingabe einer Zahl oder eines Symbols erwartet, so wird der aktuelle Wert in einem Unterfenster angezeigt. Der Schalter mit dem Label 'change' kann benutzt werden, um einen neuen Wert zu spezifizieren. Nach Anklicken dieses Schalters wird eine Dialog-Box für die Eingabe des Parameters geöffnet.



Abb. 4-4: Ein Optionen-Fenster

4.4 Menüs

Die Befehlseingabe erfolgt bei PACE über Pop-up-Menüs. Die Art des Menüs, das beim Anklicken angezeigt wird, hängt von den folgenden drei Faktoren ab:

- vom jeweiligen Fenster, in dem sich der Cursor befindet.
- davon, ob ein Element selektiert wurde oder nicht, und falls ja,
- um welches Element es sich dabei handelt.

Das entsprechende Pop-up-Menü, das eine Auflistung von Befehlen beinhaltet, erscheint an Cursorposition durch Drücken der rechten Maustaste. Hält man die Taste gedrückt und fährt man mit dem Cursor der Liste entlang auf und ab, so werden die Befehle, auf denen sich jeweils der Cursor befindet, durch einen Balken markiert. Durch Loslassen der Maustaste wird der markierte und so ausgesuchte Befehl ausgeführt. Führt man hingegen den Cursor mit der Maus aus dem Menü hinaus, so wird kein Befehl ausgeführt. Die folgende Abbildung zeigt ein Pop-up-Menü.

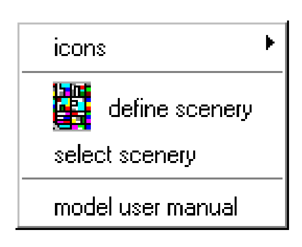


Abb. 4-5: Pop-up-Menü

4.5 Texteingabe

Für die Texteingabe stehen dem Benutzer zwei Fenster-Typen zur Verfügung. Je nach Kontext erscheint entweder das Text-Fenster oder eine Dialog-Box.

4.5.1 Text-Fenster

Text-Fenster können nur innerhalb eines übergeordneten PACE-Fenster geöffnet werden (z.B. zum Editieren von Inskriptionen) und werden gleichberechtigt mit allen anderen offenen Fenstern verwaltet. Dies bedeutet, daß man ein Text-Fenster verlassen, in einem anderen Fenster arbeiten und später wieder in das ursprüngliche Text-Fenster zurückkehren kann, um die Arbeit darin fortzusetzen.

Ein Text-Fenster verfügt auf der rechten Seite über einen Rollbalken.

Um den Inhalt eines Inskiptions-Fensters in ein PACE-Modell einzufügen, wird zunächst durch Drücken der rechten Maustaste das dem Fenster zugeordnete Pop-up-Menü aufgerufen und darin die Funktion 'accept' ausgeführt. Die weiteren Befehle des angezeigten Pop-up-Menüs sind in Abschnitt 4.5.4 beschrieben.

4.5.2 Dialogboxen

Dialogboxen erscheinen immer dann, wenn von PACE eine kurze Texteingabe benötigt und angefordert wird. Sie können erst nach erfolgter Texteingabe oder Selektion wieder verlassen werden. Abgeschlossen wird die Texteingabe entweder durch Drücken des entsprechenden Knopfes (falls vorhanden) mit der linken Maustaste oder durch Ausführen der Funktion 'accept' im Text-Editor-Menü, die wie beim Text-Fenster durch Drücken der rechten Maustaste angezeigt wird, oder durch Drücken der Return-Taste.

Um die Eingabe in eine Dialogbox zu vermeiden und ohne Eingabe wieder in die aufrufende Umgebung zurückzukehren, wird, falls

vorhanden, der entsprechende Knopf gedrückt (normalerweise ‘cancel’-Button) oder die Texteingabe mit einer leeren Dialogbox abgeschlossen.

4.5.3 Tastaturbefehle

Das Verändern der Cursorposition in Textfenstern erfolgt wie üblich mit der Maus. Außerdem stehen folgende Windows-Tastenkombinationen zur Verfügung:

Backspace	löscht das Zeichen links vom Cursor.
Entf	löscht das Zeichen rechts vom Cursor.
Strg-t	gibt die Zeichen ifTrue: an der aktuellen Cursorposition aus.
Strg-f	gibt die Zeichen ifFalse: an der aktuellen Cursorposition aus.
Strg-d	gibt das aktuelle Datum an der Cursorposition aus.
Strg-g	gibt das Zuweisungszeichen := an der Cursorposition aus.
Strg-a	Eingabe eines Strings, der gesucht werden soll.
Strg-s	Suchen des nächsten Auftreten eines eingegebenen Strings
Strg-e	Eingabe eines Strrings, der ersetzter werden soll und des Strings, der ihn ersetzen soll.
Strg-r	Erneutes ersetzen eine Strings
Strg-x	Löschen eines Strings
Strg-c	Kopieren eines Strings
Strg-v	Einsetzen eines Strings
Strg-y	Unterbechen des ablaufenden Programm
Strg-z	Letztes Kommando zurücknehmen

4.5.4 Text-Editor-Befehle

In Abhängigkeit von der Art des jeweiligen Textfenster stehen jeweils ein Teil der nachfolgend beschriebenen Text-Editor-Befehle zur Verfügung. Die jeweils verfügbaren Befehle sind im Text-Editor-Menü aufgelistet, das durch Drücken der rechten Maustaste aktiviert wird.



Abb. 4-6: Menu der rechten Maustaste in Code-Inskription-Fenstern

find
Es öffnet sich ein Fenster, in dem der Suchtext einzugeben ist. Nach Drücken des ok-Knopfs wird dieser Text ausgehend von dem aktuellen Cursor im gesamten folgenden Text gesucht. Wird der Text gefunden, so wird die Fundstelle markiert und der Cursor auf das

Ende des gefundenen Textes gesetzt. Wird er nicht gefunden, so öffnet sich ein zu quittierendes Fenster mit dem Text: "Not found".



Abb. 4-7: Fenster für das Suchen von Texten

replace
Der im "Find"-Feld einzugebende Suchtext wird durch den im "Replace"-Feld angegebenen Text ersetzt. Wird der Suchtext nicht gefunden, so wird eine Fehlermeldung ausgegeben.

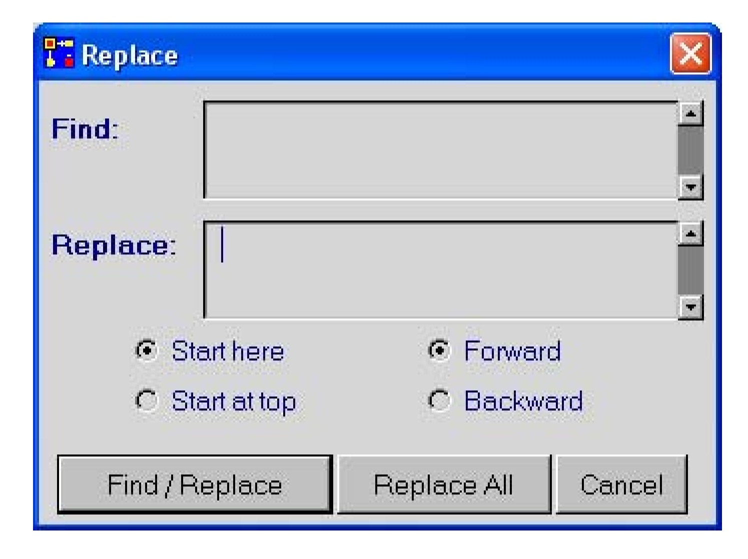


Abb. 4-8: Fenster für das Ersetzen von Texten

again
Versucht den letzten 'copy'- oder 'cut'-Befehl zu wiederholen. Bei gleichzeitigem Drücken der linken Shift-Taste wird diese Operation bis zum Textende wiederholt.

undo
Stellt den Zustand wieder her, der vor dem Ausführen des letzten Befehles vorlag.

copy
Kopiert den markierten Text in den Paste-Buffer.

cut
Löscht den markierten Text und kopiert ihn in den Paste-Buffer.

paste
Fügt den Inhalt des Paste-Buffers an der aktuellen Cursorposition ein.

do it
Falls der markierte Text aus Smalltalk-Botschaften besteht, wird er mit dem "do it"-Kmmando ausgeführt. Ergebnisse der Smalltalk-Botschaften bleiben unberücksichtigt.

print it
Falls der markierte Text aus Smalltalk-Botschaften besteht, wird er mit dem "print it"-Kmmando ausgeführt. Ergebnisse der Smalltalk-Botschaften werden direkt nach dem markierten Text ausgegeben.

inspect

Falls es sich um Smalltalk-Code handelt und der markierte Text ein Smalltalk-Objekt ist, werden dessen Charakteristika in einem Informationsfenster ausgegeben. Andernfalls wird eine Fehlermeldung angezeigt.

accept
Akzeptiert den eingetragenen Smalltalk-Code und fügt ihn in das aktuelle PACE-Modell ein. Falls kein korrekter Smalltalk-Code vorliegt, wird eine Fehlermeldung ausgegeben.

cancel
Stellt den Zustand wieder her, der nach dem Ausführen des letzten 'accept'-Befehles vorlag.

dictionary
Falls ein Text markiert ist, öffnet sich das Fenster des Modell-Lexikons. Der markierte Text ist als Schlüssel (Keyword) eingetragen. Der Anwender muß die Beschreibung (Description) zufügen. Falls das Keyword schon existiert wird eine Fehlermeldung ausgegeben.

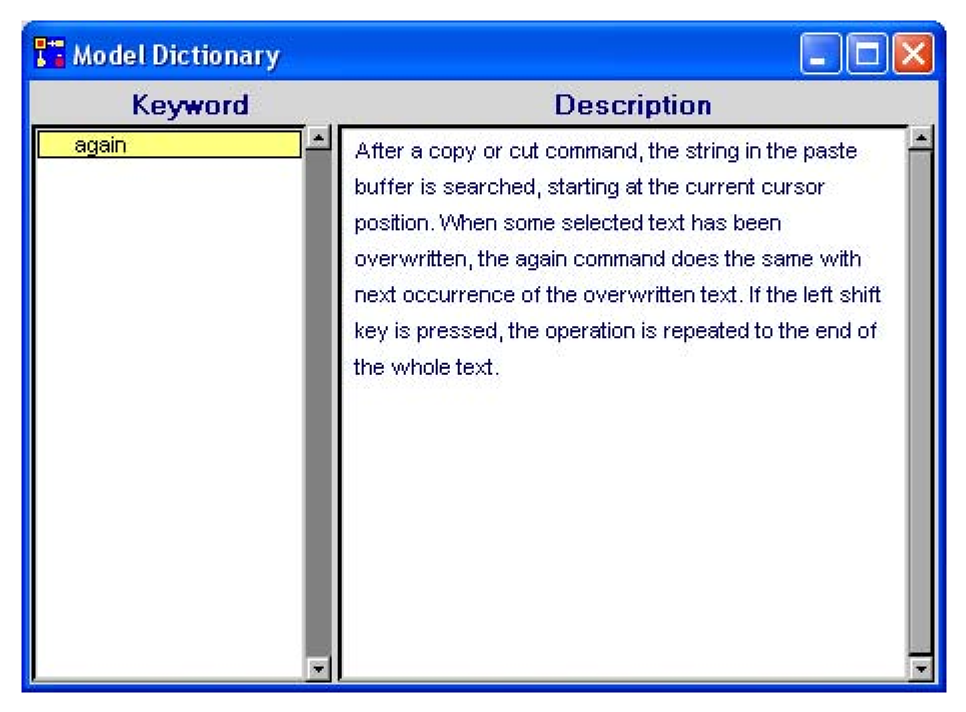


Abb. 4-9: Window of the Model-Dictionary

format
Falls es sich bei dem Text um korrekten Smalltalk-Code handelt, wird dieser in eine Standard-Form gebracht. Andernfalls wird eine Fehlermeldung ausgegeben.

hardcopy
Gibt des markierten Text auf einem Drucker aus.

...
Die Drei-Punkte-Menü-Funktion wird nur bei Netz-Inskriptionen angezeigt und ersetzt die Anzeige des Codes im Netzfenster durch drei aufeinanderfolgende Punkte. Diese Maßnahme ist bei längeren Codestücken, welche die Übersicht im Netzfenster beeinträchtigen, zweckmäßig. Man schiebt das Drei-Punkte-Ersatzsymbol möglichst in die Nähe des Netzelements zu dem es gehört (...selektieren und dann nochmals selektieren und mit gedrückter Maustatse verschieben).

Zu dem verborgenen Code gelangt man am einfachsten, indem man das Drei-Punkte-Ersatzsymbol im Edit-Mode des Netzfensters mit der linken Maustaste selektiert und danach mit der rechten Maustaste die inspect-Menüfunktion aufruft.

4.6 Ikonen

Die für die einzelnen Elemente (Stelle, Kanal, Marke, Transition, Modul) zur Darstellung gelangenden graphischen Symbole (Ikonen, Bitmaps) sind frei wählbar. Für jedes Netzelement sind jeweils zwei Ikonen defaultmäßig vordefiniert: die Standard-Ikone (standard icon) und die Alternativ-Ikone (alternate icon). Jeder Instanz aus diesen Klassen ist bekannt, ob sie mit der Standard- oder mit der Alternativ-Ikone dargestellt werden soll. Bei Erzeugung wird die Standard-Ikone verwendet. Die Ikonen können entweder aus einer vordefinierten Ikonen-Liste oder aus einer Liste von Ikonen, die vom Benutzer selbst definiert wurde, ausgewählt werden. Für jedes Netzelement kann die Ikone und ihre Darstellungsgröße individuell festgelegt werden.

Benutzer-definierte Ikonen bzw. Bitmaps werden auch für die Hintergrundbilder von Netzfenstern verwendet.

Individuelle, benutzerdefinierte Ikonen können jede beliebige rechteckige Darstellungsgröße haben und können ein beliebiges Bild

darstellen. Diese Bilder können als Bitmap eingelesen, über das Clipboard importiert und exportiert oder direkt vom Bildschirm übernommen werden. Eine individuelle Ikone wird durch ihren Namen gekennzeichnet (ein Symbol). Jedes Modell oder Unternetz kann beliebig viele individuelle Ikonen haben. All diese Ikonen können innerhalb des Modelles oder des Unternetzes eingesetzt werden. Von der PACE-Hauptleiste aus können Fenster zur Bearbeitung von Ikonen geöffnet werden.

Module können auch während der Laufzeit mit einer neuen individuellen Ikone versehen werden (Beschreibung des entsprechenden Transitions-Codes in Kapitel 3: 'Die PACE-Sprache'). Die neue Ikone ist bei der Simulation sofort zu sehen. Auf diese Weise kann der veränderte Zustand eines Moduls während der Simulation sichtbar gemacht werden.

Die Marken-Ikone und deren Name wird durch die sogenannte Ikonen-Funktion definiert. Diese wird dem Output-Konnektor einer Transition zugewiesen und legt die individuelle Ikone fest, durch die eine Marke dargestellt werden soll.

Eine detailliertere Beschreibung, wie in PACE die Ikonen eingesetzt werden, ist bei den entsprechenden Befehlen im Netz-Editor und im Simulator zu finden.

4.7 Die linke Shift-Taste

Bei der Ausführung bestimmter Funktionen steht bei gleichzeitigem Drücken der linken Maustaste und der linken Shift-Taste der Tastatur zusätzlich eine weitere Option zur Verfügung. Es folgt eine Auflistung dieser Funktionen mit der Beschreibung der jeweilige Zusatzoption.

4.7.1 Mehrfaches Anwählen

Beim Anwählen von Netz-Elementen und von Konnektoren innerhalb eines Netz-Fensters können durch Drücken der linke Shift-Taste mehrere Netzelemente zusammen selektiert werden.

Sind bereits Elemente angewählt, so kann ein weiteres Netzelement selektiert werden, indem das Element mit der linken Maustaste angeklickt und dabei gleichzeitig die linke Shift-Taste gedrückt wird.

Wird beim Anklicken eines weiteren Elements die linke Shift-Taste nicht gedrückt, so werden die bereits ausgeführten Selektierungen wieder zurückgesetzt. Für mehrfaches Anwählen muß daher immer die linke Shift-Taste verwendet werden.

Liegt der Text von Inskriptionen über den auszuwählenden Netz-Elementen, so wird dieser nicht ausgewählt, wenn bei der Selektion des Netz-Elementes die linke Shift-Taste gedrückt wird.

4.7.2 Festlegen der Fenstergröße beim Öffnen

Bestimmte Fenster (z.B. Marken-Fenster) werden mit einer vordefinierten Standardgröße an der Cursor-Position geöffnet. Der Anwender kann entweder nachträglich unter Verwendung von Funktionen des unterliegenden Fenster-Systems das Fenster verschieben und

vergrößern oder unter Verwendung der linken Shift-Taste schon beim Öffnen eine andere Fenstergröße festlegen.

Wird beim Öffnen des Fensters die linke Shift-Taste gedrückt, so wird ein rechteckiger Rahmen in der vordefinierten Standardgröße mit dem Cursor im linken oberen Eck angezeigt. Durch Verschieben der Maus kann das Rechteck simultan auf dem Bildschirm verschoben werden. Hat man die linke obere Ecke des Rechtecks an die gewünschte Stelle platziert, so wird die linke Maus-Taste gedrückt und gedrückt gehalten. Der Cursor springt in die untere rechte Ecke des Rechtecks. Die Maus wird bei gedrückter linker Maus-Taste solange verschoben, bis das Rechteck die gewünschte Größe hat. Nach Loslassen der linken Maus-Taste wird die Größe des Fensters eingefroren.

4.7.3 Verschieben von Netz-Elementen

Mehrere selektierte Netz-Elemente können durch Drücken der linken Shift-Taste gemeinsam verschoben werden. Im einzelnen ist dabei wie folgt vorzugehen:

1. Selektieren eines oder mehrerer Netzelemente.

2. Gleichzeitig die linke Shift-Taste drücken, den Cursor auf eines der selektierten Elemente positionieren und die linke Maustaste drücken.

3. Wenn Sie nun die linke Shift-Taste loslassen, erscheint ein Rechteck, das die selektierten Netz-Elemente umrahmt. Dieses Rechteck können sie mit der weiterhin gedrückten linken Maustaste in dem Fenster herumschieben.

4. Wenn Sie dabei die linke Shift-Taste erneut drücken, so sehen Sie die selektierten Netz-Elemente einschließlich der Konnektoren während des Verschiebens, also die sich durch das Verschieben ändernde Netz-Topologie

5. Diese wird eingefroren, sobald Sie die linke Maustaste loslassen.

4.7.4 Anwahl zurücksetzen

Ist mindestens ein Element innerhalb eines Netz-Fensters angewählt und befindet sich der Cursor innerhalb des Netz-Fensters, so setzt das gleichzeitige Drücken der linken Shift-Taste und der Ctrl-Taste alle Selektierungen im aktuellen Fenster wieder zurück.

4.7.5 Fenster wieder aufbauen

Ist kein Element angewählt, baut das gleichzeitige Drücken der linken Shift-Taste und der Ctrl-Taste das Fenster wieder neu auf (refresh).

4.7.6 Der 'find'-Befehl im Manual

Durch Niederdrücken der linken Shift-Taste beim Ausführen des 'find'-Befehls im Online-Manual wird die angegebene Zeichenfolge nicht nur in den Überschriften sondern auch in den Texten der einzelnen Einträge gesucht.

4.8 Unterbrechen im Notfall

Falls ein Modell bei Fehlern in Inskriptionen in eine Schleife gerät, kann es häufig nur durch Eingreifen des Anwenders angehalten werden. Eine unbedingte Unterbechung kann durch gemeinsames Drücken der Taste Strg und der Taste für den Buchstaben y (bekannt unter 'Ctrl-y')[6] erreicht werden. Nach Quittieren des sich öffnenden Abfragefensters kann die Arbeit mit dem Modell fortgesetzt werden.

[6] Ctrl ist die englische Bezeichnung der Strg-Taste

5 BEDIENLEISTEN

5.1 Die PACE-Hauptleiste

Die PACE-Hauptleiste ist das erste bedienbare PACE-Fenster, das erscheint, nachdem PACE installiert und aufgerufen wurde. Über die PACE-Hauptleiste sind sämtliche PACE-Funktionen direkt oder indirekt erreichbar. Die PACE-Hauptleiste ist bei der Modellentwicklung immer geöffnet und befindet sich normalerweise in der linken oberen Ecke des Bildschirmes. Sie kann aber auch an einer beliebigen anderen Stelle des Bildschirms positioniert werden. Durch Auswahl eines Menüpunkts der Leiste mit der linken Maustaste wird das zugehörige Menü angezeigt. Einige häufig benötigte Menüfunktionen können durch Drücken der unter den Menüpunkten angeordneten bebilderten Druckknöpfe mit der linken Maustaste direkt ausgeführt werden.

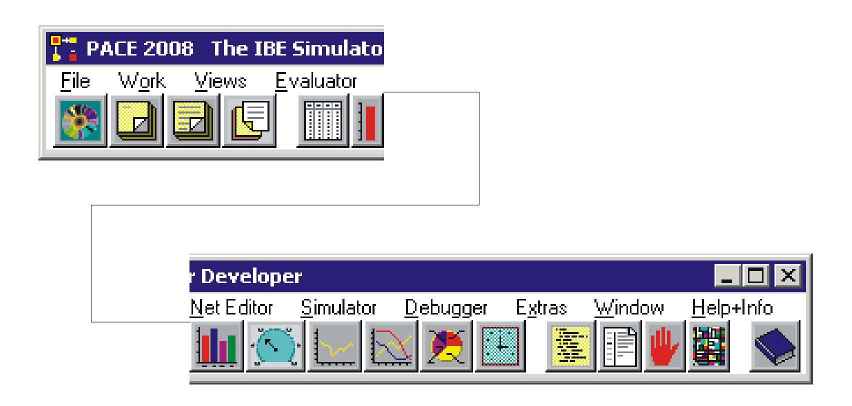


Abb. 5-1: Die PACE 2008 Hauptleiste

5.2 File-Menü

Beim ersten Starten von PACE nach der Installation wird ein leeres Image geladen. Es besteht nur aus der PACE-Hauptleiste (Abb. 5-1) und enthält noch keinerlei Netz- Informationen.



Abb. 5-2: File-Menü

Netze können danach auf zwei verschiedene Weisen eingebracht werden, nämlich durch Laden eines anderen Images, in dem schon ein Netz geladen ist, oder durch Erstellen oder Laden einer Netzbeschreibung, kurz eines Netzes oder Modells.

Entsprechend kann man auch beim Speichern verfahren. Man kann einerseits das geladene Netz mit der gesamten PACE-Umgebung als Image abspeichern. Beim erneuten Laden des abgespeicherten Images erhält man dann das Netz einschließlich Umgebung genau so zurück, wie es vor dem Speichern vorlag. Die Verwendung von Images ist also für den täglichen Betrieb angezeigt, wenn man an einem Modell arbeitet oder wenn ein fertiges Modell ausgeführt wird.

Die zweite Möglichkeit, Netze abzuspeichern, ist vorzugsweise bei der Archivierung von Netzen, bei der Portierung von Netzen auf andere Betriebssysteme oder neuere PACE-Versionen und bei Modulen zu verwenden. Dabei werden Netze bzw. Module in Form einer Netzbeschreibung in dem PACE-Verzeichnis ‘nets’ bzw. ‘modules’ abgelegt. Diese Methode hat den Vorteil, daß für gespeicherte Netze vergleichsweise wenig Speicherplatz erforderlich ist. Sie hat aber den großen Nachteil, daß die Informationen über die Darstellung des Netzes auf dem Bildschirm und über die an das Netz angeschlossenen Visualisierungs- und Diagrammfenster teilweise verloren gehen. Nach dem erneuten Laden des Netzes muss die Darstellung auf dem Bildschirm wieder manuell aufgebaut werden.

5.2.1 Laden und Speichern von Images

5.2.1.1 load image

Die angegebene Funktion (siehe Abb. 5-2) startet ein File-Auswahlmenü (siehe Abb. 5.3), mit dem das zu ladende Image ausgewählt wird.

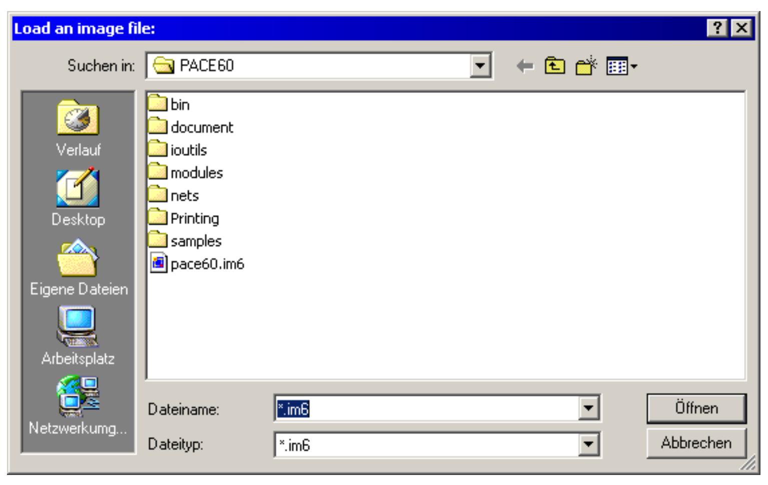


Abb. 5-3: Auswahl eines Images

Das laufende Verzeichnis von PACE (current oder default directory) wird auf das Verzeichnis eingestellt, aus dem das Image geladen wird.

5.2.1.2 store image

Ein Image sollte nur im laufenden Verzeichnis gespeichert werden, weil normalerweise in den PACE-Unterverzeichnissen zugeordnete Informationen abgespeichert sind. Z.B. kann der Anwender in Unterverzeichnis 'ioutils' die Parameter von Visualisierungsfenstern und Diagrammen abspeichern.

Die Funktion öffnet ein Fenster, in dem der Name des zuletzt geladenen Images angezeigt wird.

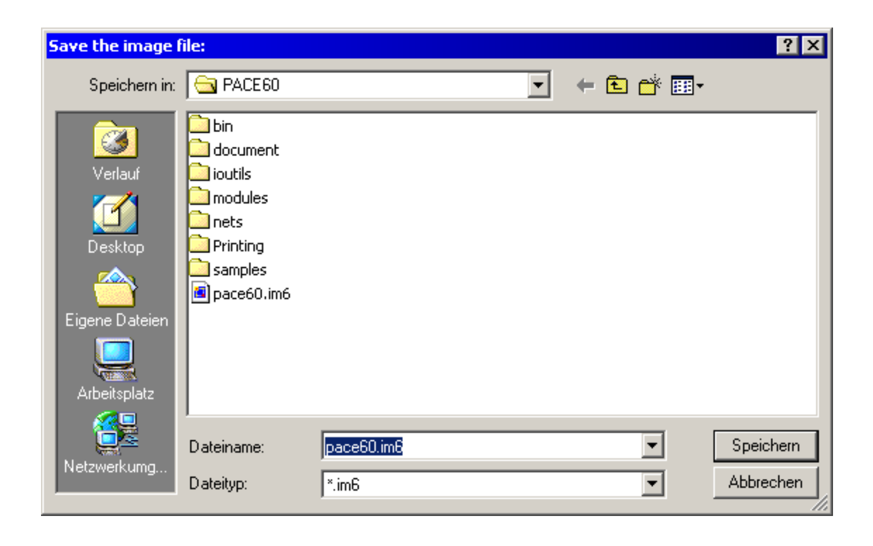


Abb. 5-4: Eingabe-Fenster für den Image-Namen

5.2.2 Handhabung von Modellen

5.2.2.1 new net

Es öffnet sich ein Eingabefenster, in dem der Default-Name 'newNet' angezeigt wird. Dieser wird durch den Name des neuen Modells ersetzt. Soll der Modellname später geändert werden, so kann der neue Name beim Speichern des Modells angegeben werden (siehe Abschnitt: 'store net').

Nach Drücken der Return-Taste öffnet sich ein Fenster mit einer Netz-Liste, in der nur der gewählte Modellname angezeigt wird. Es kann jetzt mit Entwicklung des neuen Modells begonnen werden. Dazu wird die Zeile, in der der Netzname steht, mit der linken Maustaste selektiert. Die weitere Bearbeitung ist identisch mit der Bearbeitung eines schon erstellten Modells (siehe 'load net').

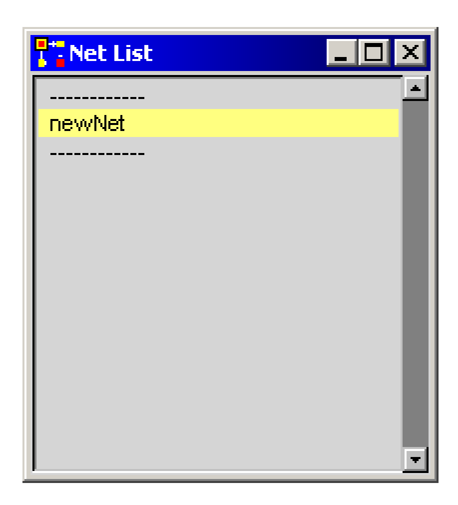


Abb. 5-5: Erstellen eines neuen Modells

5.2.2.2 load net

Zunächst wird ein Auswahlfenster mit sämtlichen Netzen, die im ausgewählten Verzeichnis 'nets' des aktuellen PACE-Verzeichnisses gespeichert sind, geöffnet (siehe Abb. 5-6). Das gewünschte Modell wird durch Selektieren des zugeordneten Elements in dieser Liste ausgewählt und durch Drücken des Knopfs ‘Öffnen’ geladen. Enthält das Modell Ikonen, so werden diese ebenfalls im ausgewählten ‘nets’- Verzeichnis erwartet. Die Ikon-Datei hat den gleichen Name wie das Netz, aber die Extension 'icn'.

Nach dem Laden des Modells wird ein Fenster mit der sog. Netzliste des Modells angezeigt (siehe Abb. 5-7). Das Arbeiten mit PACE, das heißt das Editieren oder die Simulation von PACE-Modellen, wird von diesem Fenster aus begonnen.

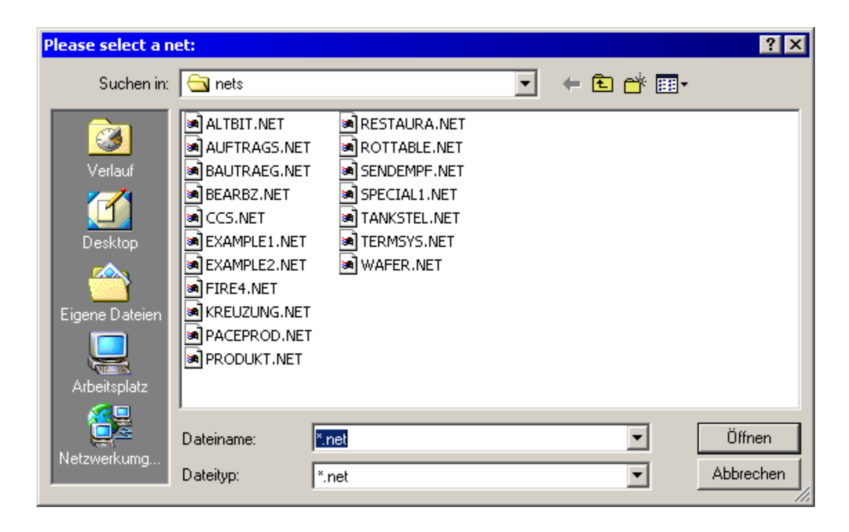


Abb. 5-6: Auswahlmenü für das Laden eines Netzes

In der Netzliste wird die Hierarchie-Struktur der angezeigten Module durch Einrücken der Modulnamen kenntlich gemacht. Von diesem Netz-Listen-Fenster aus können Netzfenster in jeder Hierarchie-Stufe eines PACE-Modells geöffnet werden. Das Netz-Listen-

Fenster bleibt während der gesamten Arbeit an bzw. mit einem Modell geöffnet.

Ein Netz oder Teilnetz wird durch durch Anklicken einer Zeile der Netzliste mit der linken Maustaste angewählt. Wird danach die rechte Maustaste gedrückt, so wird das in Abb. 5-7 dargestellte Menü angezeigt.

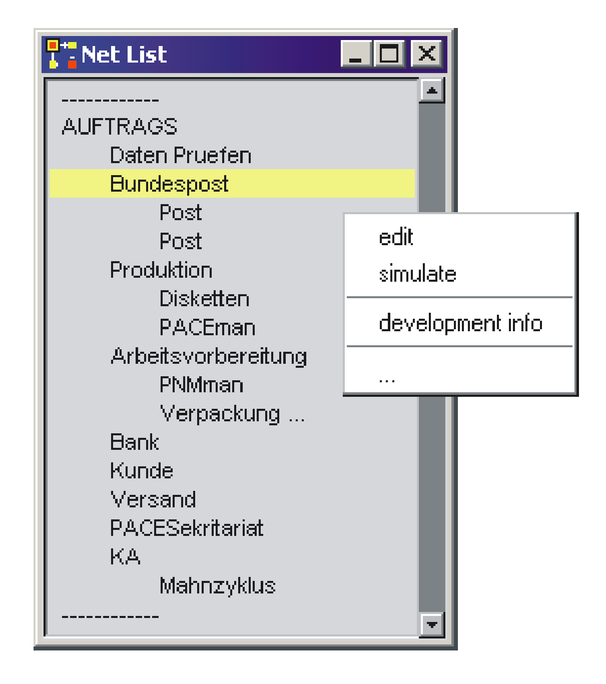


Abb. 5-7: Netzliste mit Auswahlmenü

edit
Öffnet oder aktiviert für das angewählte Netz ein Netz-Fenster im Editier-Modus. Darin wird das Netz angezeigt und kann mittels des in PACE integrierten Graphik-Editors bearbeitet werden.

simulate
Öffnet oder aktiviert für das angewählte Netz ein Netz-Fenster im Simulations-Modus. Darin kann das dynamische Verhalten des

Netzes durch Ausführen des Netzes wahlweise mit oder ohne Animation des Markenflusses beobachtet werden.

development info
Öffnet ein Text-Fenster, in dem zu jedem Modul ein beliebig langer Text über die Entwicklung des Moduls geschrieben werden kann. Die Entwicklungsinformation sollte mindestens die Entwicklungs- und ggf. Weiterentwicklungs-Zeiträume und die jeweiligen Bearbeiter des Moduls angeben.

'development info' ist der Modulliste zugeordnet und sollte nur organisatorische und keine technischen Informationen über den Modul enthalten. Letztere werden beim Modul selbst in der 'purpose description' niedergelegt.

Zur Dokumentation des Modells werden sowohl die 'development info' als auch die 'purpose description' zusammen mit der graphischen Darstellung des Moduls und weiteren Moduldaten (z.B. Netzvariable) über 'print all' oder 'print selected net' (siehe den Menüpunkt 'print' im file-Menü) ausgedruckt.

...
Diese Funktion dient zwei Zwecken:

- Erstens ermöglicht sie bei großen Netzen eine einfachere Handhabung der Netz-Liste. Mit dieser Funktion kann von einer Hierarchie-Ebene aus die Anzeige untergeordneter Netze in der Netz-Liste unterdrückt werden. Wurde ein Unternetz damit ausgeschaltet, folgen seinem Namen in der Netz-Liste drei Punkte (...). Die Funktion hat eine Schalterfunktion: ausblendete Unternetze können durch erneutes Ausführen der Funktion wieder eingeblendet werden.

- Zweitens unterbindet diese Funktion von einem nicht gebundenen ('non-fixed') Simulations-Fenster aus das Aktivieren von Fenstern, indem das angewählte Netz oder eines seiner hierarchisch untergeordneten Netze dargestellt ist. Dazu ist die entsprechende Option im Menü ‘Simulator’ einzuschalten.

5.2.2.3 store net

Der 'store net'-Befehl speichert das gesamte Modell als Netz in einem File.

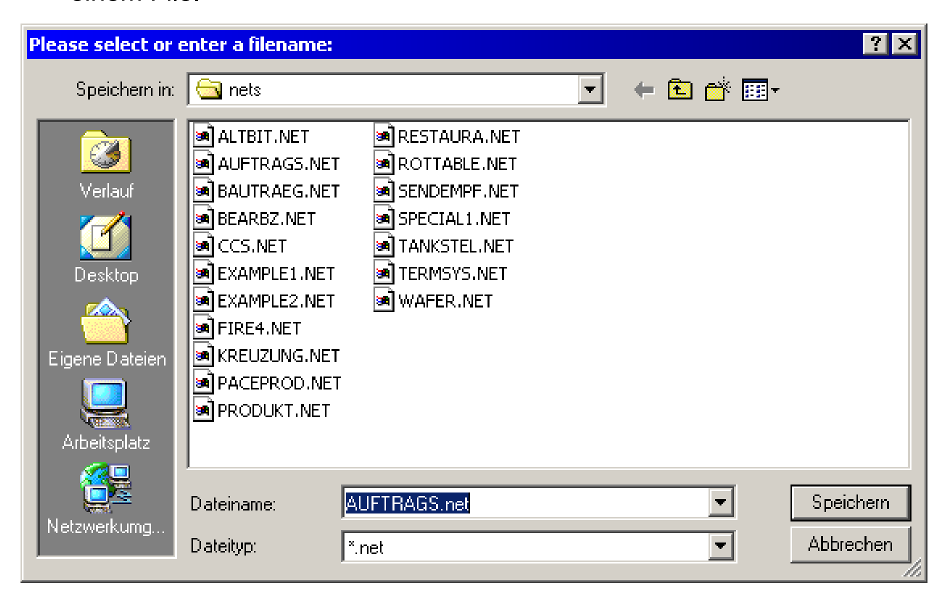


Abb. 5-8: Speichern eines Netzes

Beim Speichern eines Modelles wird der alte Filenamen vorgeschlagen. Der Anwender kann einen neuen Namen eingeben.

Der Filename wird aus dem vom Benutzer anzugebenden Modellnamen und der Erweiterung (extension) '.net' gebildet. Das so erstellte File wird standardmäßig im Unterverzeichnis 'nets' des PACE-Verzeichnisses abgelegt, sofern der Anwender kein anderes Verzeichnis vorgibt. Enthält das Model Ikonen, so werden diese ebenfalls im Netzverzeichnis gespeichert. Die Ikon-Datei hat den gleichen Namen wie das Netz, aber die Erweiterung 'icn'. Weitere Dateien, die im Netzverzeichnis gespeichert werden, sind zwei Dateien mit dem Netznamen und den Erweiterungen ‘.imd’ und ‘.mdc’. Diese enthalten das Modellhandbuch (model description), das über den

Menüpunkt ‘model user manual’ im Support-Menü erstellt wird, und das Model-Lexikon (model dictionary), das im Menü ‘Net Editor’ aufrufbar ist.

5.2.2.4 store module

Der selektierte Modul und dessen Submodule werden einschließlich der das Interface bildenden S--Elemente in einem File gespeichert.

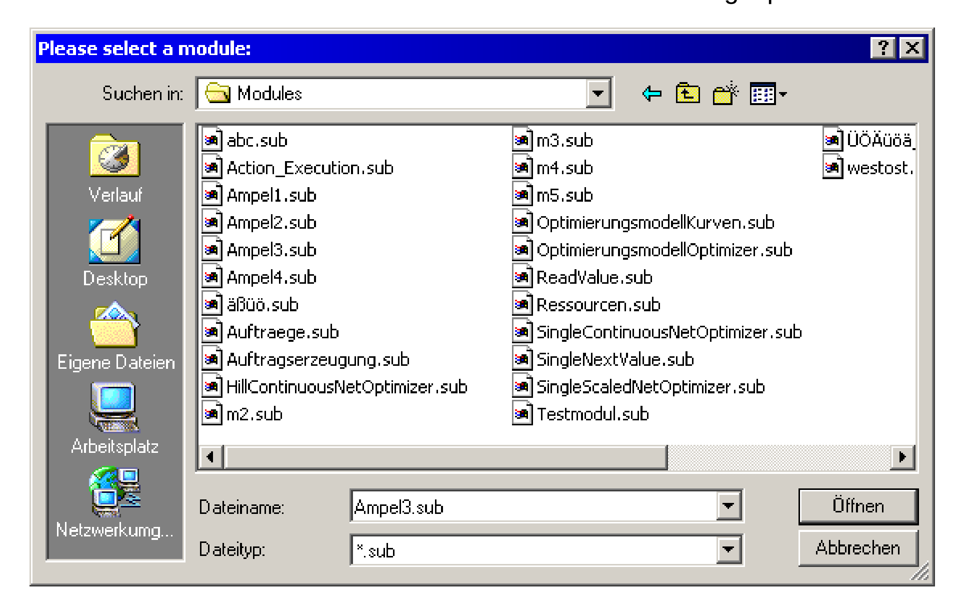


Abb. 5-9: Speichern eines Moduls

Der Name dieses File wird aus dem Namen des Moduls und der Erweiterung '.sub' gebildet, die von PACE hinzugefügt wird. Der Benutzer kann einen neuen Namen angeben. Das File wird im Unterverzeichnis 'modules' des PACE-Verzeichnisses gespeichert, sofern der Anwender kein anderes Verzeichnis vorgibt. Ikonen des Moduls werden ebenfalls in diesem Verzeichnis gespeichert. Das Ikon-File hat den Modul-Namen mit der Erweiterung 'icn'.

Ein so gespeichertes Modul kann zu jedem Modell hinzugefügt werden. Dies geschieht durch Ausführen der 'restore module'-Funktion in einem zum Editieren geöffneten Netz-Fenster (siehe Kapitel 7).

Module ermöglichen die Erstellung von Standard-Komponenten für unterschiedliche Aufgabengebiete und unterstützen damit den komponentenweisen Aufbau von Simulationsmodellen.

5.2.2.5 leave net

Die Anweisung schließt alle über die Netzliste oder deren Unterfenster geöffneten Fenster und schließlich die Netzliste selbst. Danach kann ein weiteres Modell erstellt oder geladen werden.

5.2.3 Drucken eines Modells

Das Print-Menü dient zur Ausgabe der System-Dokumentation. Die Dokumente werden im PostScript-Format ausgegeben und können mit einem geeigneten PC-Tool (z.B. GSview) angezeigt und ins PDF-Format gewandelt werden, in dem auch die Hauptteil der übrigen PACE-Dokumentation vorliegt.

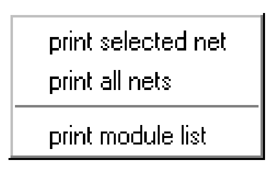


Abb. 5-10: Print-Menü

Für die Erstellung der Nutzer-Dokumentation empfehlen wir die Verwendung eines gängigen Textverarbeitungstools. PACE-Fenster werden mit einem Graphik-Programm, welches das Ausführen von Screen-Shots unterstützt (z.B. Corel Draw, Paint Shop Pro), fotografiert, in einem gängigen Graphikformat abgespeichert und direkt in das Textdokument eingesetzt.

5.2.4 Druck-Funktionen

print all nets
Generiert eine PostScript-Datei, die alle Netze einschließlich aller Inskriptionen und Kommentare enthält. Der Name dieser Datei wird aus dem Modellnamen mit der Erweiterung '.ps' gebildet und wird im Unterverzeichnis 'printing' des PACE-Verzeichnisses gespeichert.

print selected net
Mit diesem Befehl kann eine PostScript-Datei für das in der Netzliste selektierte Teilnetz erzeugt werden. Der Name der Datei wird aus dem Netznamen mit der Erweiterung '.ps' gebildet und wird im Unterverzeichnis 'printing' gespeichert.

print module list
Die Netz-Liste wird als PostScript-Datei ausgegeben. Der Name der Datei wird aus dem Modellnamen mit der Erweiterung 'modulelist.ps' gebildet und wird im Unterverzeichnis 'printing' gespeichert.

5.2.5 Druck-Optionen

Die Funktion ‘options’ öffnet ein Fenster, in dem Parameter für die Gestaltung des Ausdrucks gesetzt werden können.

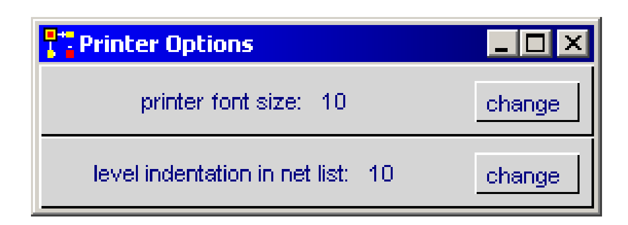


Abb. 5-11: Print-Optionen

Die gesetzten Optionen, sind für das gesamte geladene Netz gültig. Die einzelnen Parameter-Einstellungen werden zusammen mit dem Modell geladen und gespeichert.

printer font size
Definiert die Größe der Schrift, welche bei einem Ausdruck des angewählten Modells benutzt werden soll.

level indentation in net list
Setzt die Anzahl Zwischenräume, die für das Einrücken der nächst niedrigeren Hierarchiestufe in der Netz-Liste verwendet werden sollen.

5.2.6 Beenden von PACE

Über den Menüpunkt ‘leave PACE’ kann PACE beendet werden.

Damit nicht versehentlich schon geleistete Modellierungsarbeit verloren geht, wird vor dem eigentlichen Verlassen des Programs ein Menü angezeigt, das dem Benutzer das Retten des vorliegenden Images nahelegt.

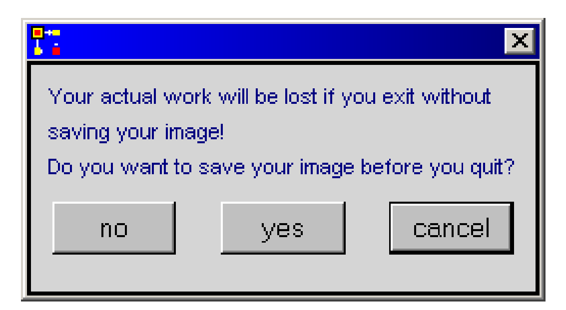


Abb. 5-12: Verlassen von PACE

Das Speichern erfolgt wie in Abschnitt 5.2.1.2 beschrieben.

5.2.7 Arbeitsverzeichnisse

Der Menüpunkt ‘work directories’ öffnet ein Fenster, in dem die Voreinstellungen für die verschiedenen PACE-Arbeitsverzeichnisse festgelegt werden können. Wie in Kap. 2 beschrieben, handelt es sich dabei normalerweise um Unterverzeichnisse des

PACE-Verzeichnisses. Falls ein benutzerdefiniertes Verzeichnis nicht mehr existiert, wird wieder das Default-Verzeichnis verwendet.

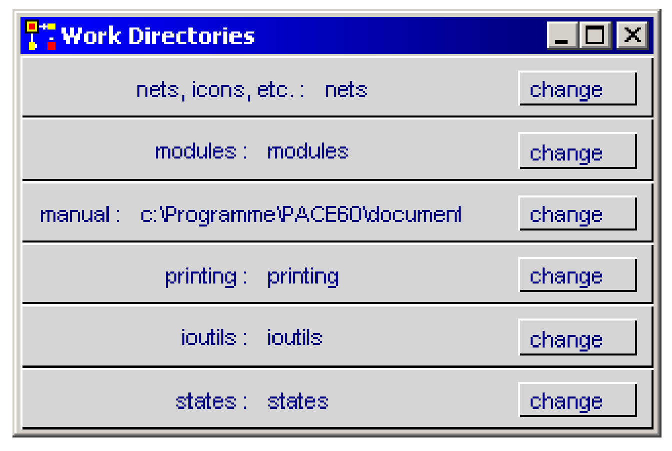


Abb. 5-13: Festlegen von Arbeitsverzeichnissen

Nach Drücken eines ‘change’-Druckknopfs öffnet sich ein Auswahlfenster für das betreffende Verzeichnis (Abb. 5-14).



Abb. 5-14: Auswahlfenster für Arbeitsverzeichnisse

5.3 Work-Menü

Über das Work-Menü sind allgemeine Hilfsfunktionen aufrufbar, die während der Arbeit mit PACE fallweise benötigt werden.

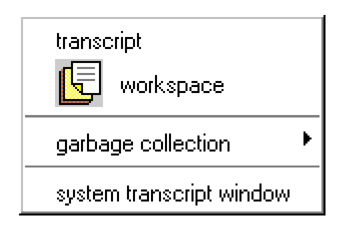


Abb. 5-15: Work-Menü

5.3.1 Mitteilungsfenster (Transcript)

Durch das Ausführen der Funktion 'transcript' wird das sog. Transcript-Fenster geöffnet, in das allgemeine Systemmeldungen ausgegeben werden. Das Fenster kann auch von PACE verwendet werden. In alle PACE-Inskriptionen können Ausgabebefehle eingefügt werden, die beispielsweise Meldungen über den Verlauf der Simulation in das Transcript-Fenster schreiben.

Auf Transcript wird durch die globale Variable 'Transcript' verwiesen. Folgendes Statement lässt eine Meldung in allen geöffneten Transcript-Fenstern erscheinen:

```
Transcript show: 'Hallo PACE-Benutzer !'.
```

Sobald ein Transcript-Fenster geschlossen wird, geht sein Inhalt verloren.

Zur Formatierung des Textes in Transcript-Fenstern sind die folgenden Botschaften vorgesehen:

tab	Tabulatorzeichen
tab: anInteger	eine definierte Anzahl von Tabulatorzeichen
cr	Wagenrücklauf
crtab	Wagenrücklauf mit nachfolgendem Tabulator
crtab: anInteger	Wagenrücklauf mit vorgegebener Anzahl von Tabulatorzeichen

Beispiel:

```
Transcript crtab; show: 'Hallo PACE-Benutzer !'.
```

5.3.2 Arbeitsfenster (Workspace)

Die Menüfunktion ‘workspace’ öffnet ein Arbeitsfenster, in dem Smalltalk-Code direkt ausgeführt werden kann. Das Fenster kann beispielsweise auch benutzt werden, um irgend einen Text für eine gewisse Zeit aufzuheben.

Sobald ein Workspace geschlossen wird, geht sein Inhalt verloren.

5.3.3 Garbage collection

5.3.3.1 Größe des aktuell verwendeten Speichers

Mit den Funktionen ‘memory size’ und ‘free memory’ kann die Größe des aktuell verwendeten und die des schon allokierten aber noch freien Speichers angezeigt werden.

PACE allokiert dynamisch je nach Bedarf freien Speicher und gibt diesen später, wenn er nicht mehr benötigt wird, wieder frei.

5.3.3.2 Garbage Collector

Mit dem Menüpunkt ‘collect garbage’ wird der System-Speicher aller Objekte freigegeben, die vom System nicht mehr benötigt werden. Diese Speicherbereinigung (englisch: garbage collection) wird vom System normalerweise automatisch ausgeführt.

Der manuelle Start der Speicherbereinigung ist in folgenden Fällen angezeigt:

- nach dem Schließen von überflüssigen Fenstern oder nach anderen Bereinigungstätigkeiten.

- nach Abbruch eines speicher-intensiven Ablaufs. Falls ein Fehler-Fenster angezeigt wird, ist dieses zuerst zu schließen, bevor die Speicherbereinigung aufgerufen wird.

- bevor eine speicher-intensive Funktion ausgeführt wird.

Wieviel Speicher durch nicht mehr benötigte Objekte belegt wurde und wieviel Speicher nach der Speicherbereinigung verfügbar ist, wird in das oder die geöffneten Transcript-Fenster ausgegeben.

5.4 View-Menü

Beim View-Menü handelt es sich um ein recht umfangreiches Menü, in dem alle zeitunabhängigen PACE-Visualisierungs- und PACE-Diagramm-Objekte anwählbar sind. Weiter ist die Festlegung von Farben für zahlreiche Fenster- und PACE-Komponenten möglich.

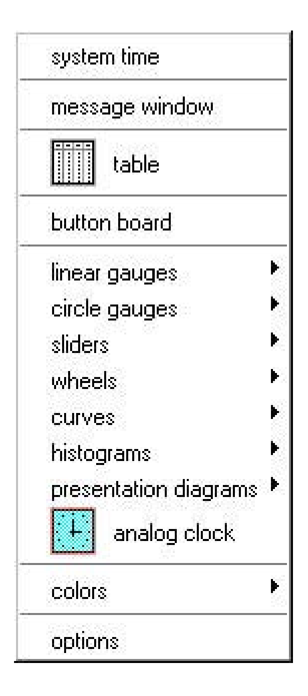


Abb. 5-16:View-Menü

Alle PACE-Ein/Ausgabe, -Visualisierungs- und -Diagramm-Objekte werden gemeinsam in Kap. 6 beschrieben und deshalb hier nicht weiter erläutert.

5.4.1 Systemzeit

Die Funktion 'system time' öffnet ein Fenster, in dem die Simulationszeit angezeigt wird. Es handelt sich dabei um den Wert der globalen Variablen 'CurrentTime', der zu Beginn eines Simulationslaufs gleich Null gesetzt wird.



Abb. 5-17:Time-Window

Der Anwender legt bei der Erstellung eines Modells (gedanklich) fest, welcher realen Zeitspanne eine Einheit der Systemzeit zugeordnet sein soll. Aus dieser Zuordnung lassen sich dann alle weiteren Zeiteinheiten in Inscriptionen ableiten.

5.4.2 Verfügbare Farben

Der Menüpunkt ‘default platform colors’ zeigt die in Abb. 5-18 dargestellte Liste der für die Colorierung von Netzkomponenten zur Verfügung stehenden Farben. Die Funktion gibt die Namen der Farben aus und zeigt, wie diese auf dem Bildschirm aussehen.

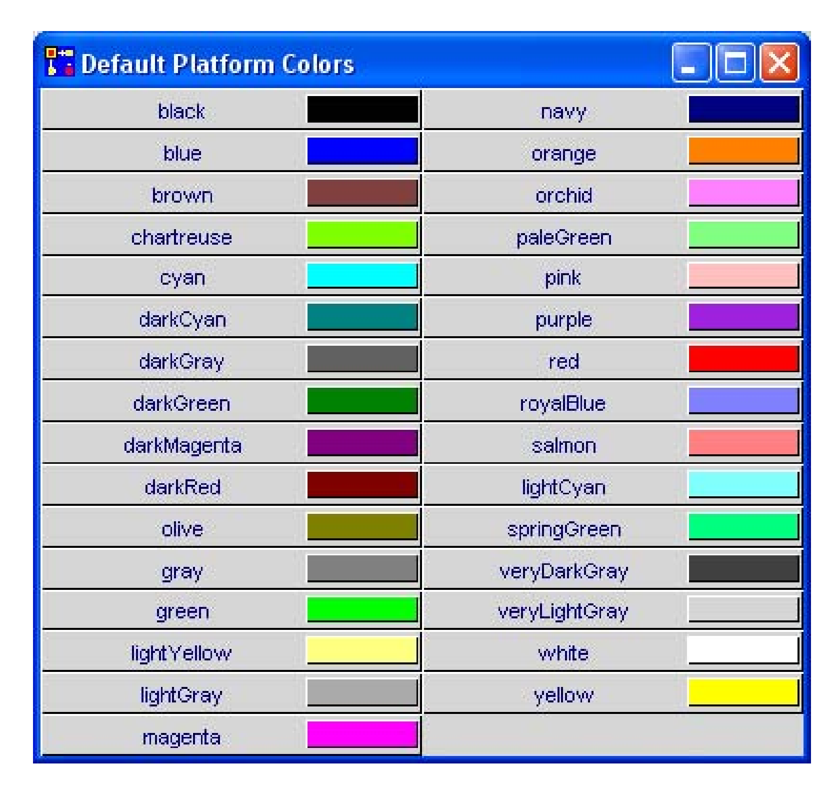


Abb. 5-18:Liste der zur Verfügung stehenden Farben

5.4.3 Farben von Fenster-Komponenten

Nach Anwahl des Menüpunkts ‘windows colors’ wird das in Abb. 5.19 dargestellte Auswahlfenster angezeigt. Mit ihm können die Farben verschiedener Fenster-Komponenten neu festgelegt werden.

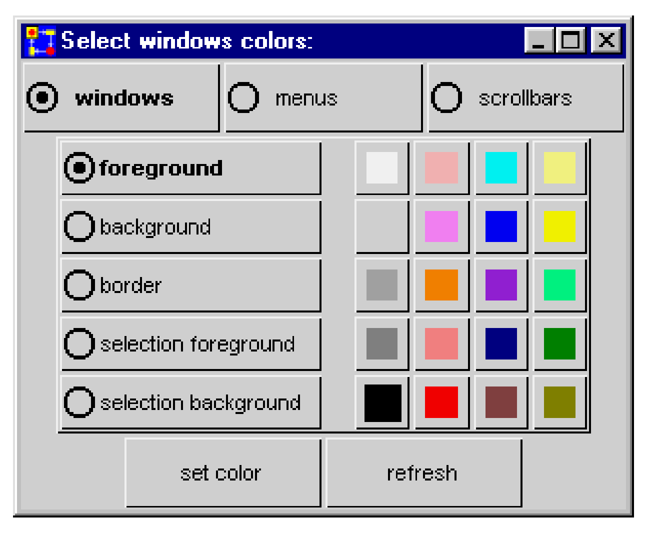


Abb. 5-19: Auswahlfenster für Farben von Fenster-Komponenten

5.4.4 Farben von Netz-Komponenten

Über den Menüpunkt ‘net constituents colors’ können die Farben für die Komponenten der verschiedenen Elemente von PACE-Netzen vorgegeben werden.

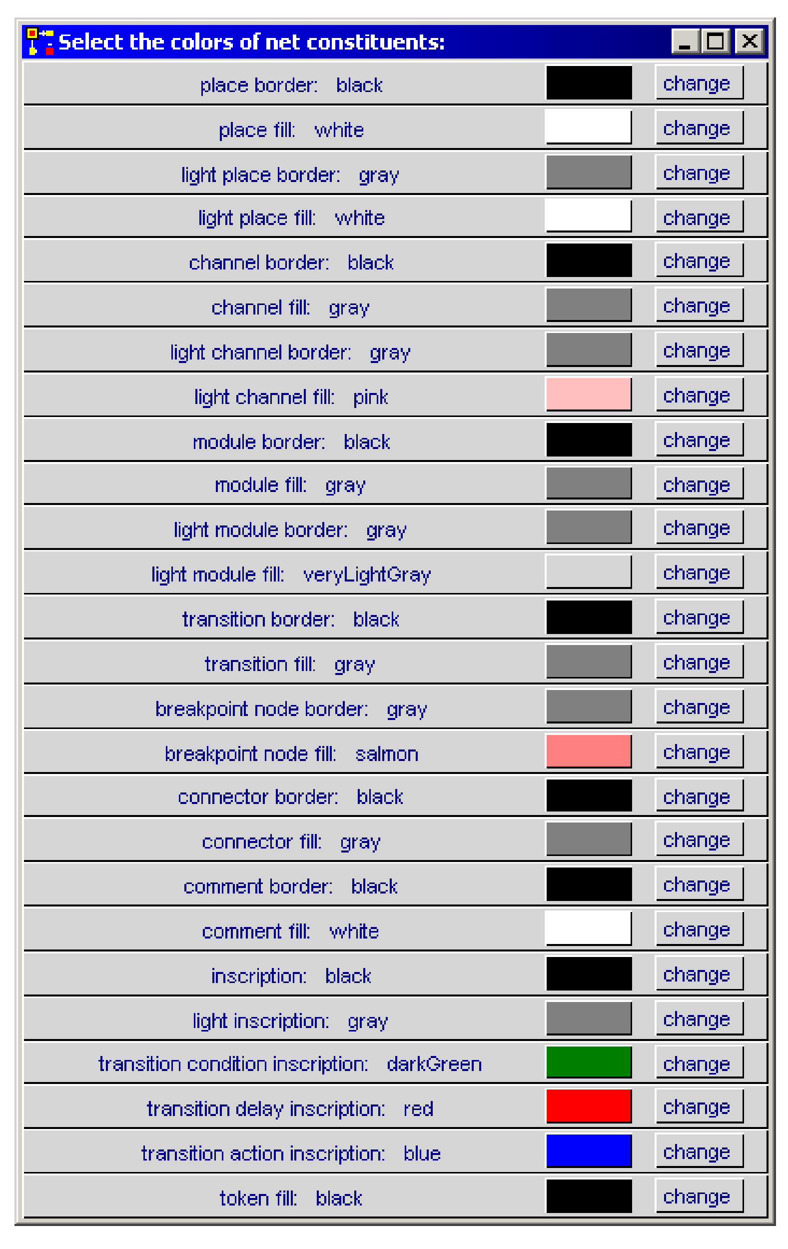


Abb. 5-20: Vom Benutzer einstellbare Farben für die verschiedenen graphischen Elemente

5.4.5 View-Optionen

Bei Ausführen der Option-Funktion im ‘view’-Menü öffnet sich das in Abb. 5-21 dargestellte Fenster.

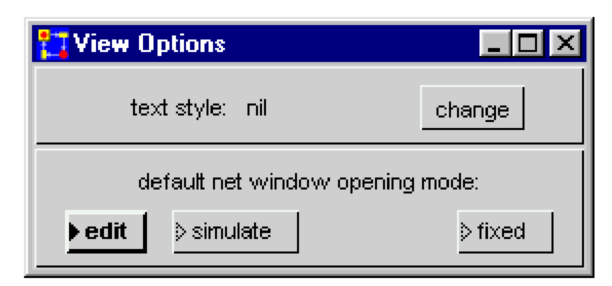


Abb. 5-21: View-Optionen-Fenster

Nach Drücken des ‘change’-Knopfes kann über das in Abb. 5-22 dargestellte Auswahlfenster der Textstil für PACE festgelegt werden.



Abb. 5-22: Auswahlfenster zur Festlegung des Textstil

Das zweite Feld ‘default net window opening mode’ eröffnet die Möglichkeit, die Vorbesetzungen der Schalter zu bestimmen, mit denen ein Netzfenster standardmäßig geöffnet werden soll. Diese Festlegung wird bei den Funktionen zum Öffnen eines Netzfensters, die den Öffnungsmodus jeweils explizite festgelegen, nicht ausgewertet.

5.5 Evaluator-Menü

Das Evaluator-Menü wurde eingeführt, damit sich der Anwender vor dem Einsatz bestimmter PACE-Features eine Vorstellung von deren Verhalten und von den Konsequenzen ihres Einsatzes machen kann.

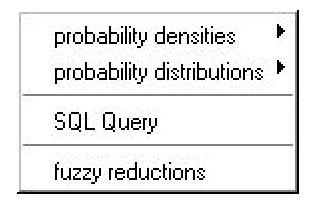


Abb. 5-23: Evaluator-Menü

5.5.1 Mathematische Wahrscheinlichkeiten

Bei Auswahl eines der probability-Menüpunkte und dann einer der mathematischen Verteilungen öffnet sich ein Fenster, in das der Anwender die Parameter der zu zeichnenden Verteilung eingeben kann. Für eine Weibull-Verteilung sieht das z.B. wie folgt aus:

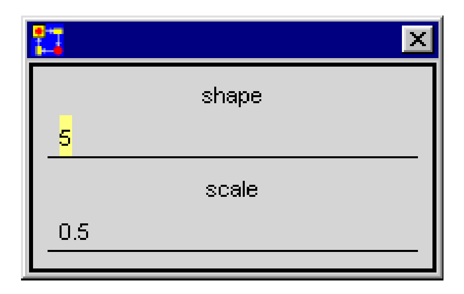


Abb. 5-24: Parameter-Eingabefenster für eine Weibull-Verteilung

Nach Drücken der Return-Taste wird je nach Menüpunkt entweder die Dichtefunktion oder die Verteilungsfunktion angezeigt:

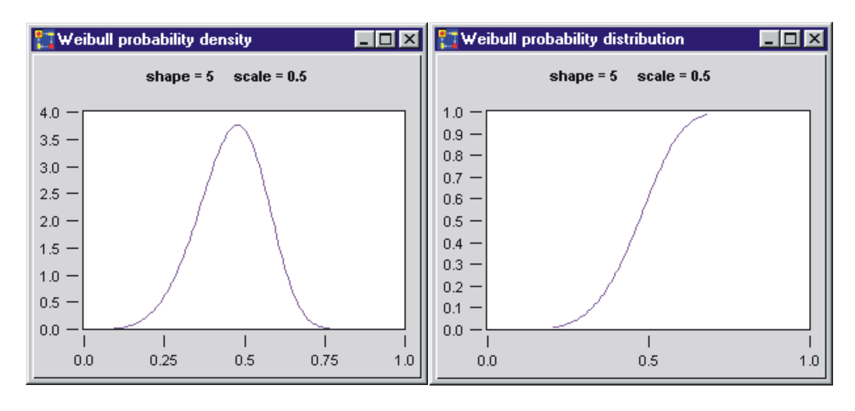


Abb. 5-25: Eine Weibull-Dichte und -Verteilung

5.5.2 Empirische Wahrscheinlichkeiten

Auswahl eines der beiden probability-Menüpunkte und dann der Menüfunktion 'empirical'. Es öffnet sich ein Fenster, in dem die verfügbaren empirischen Verteilungsfunktionen aufgelistet sind.[7].

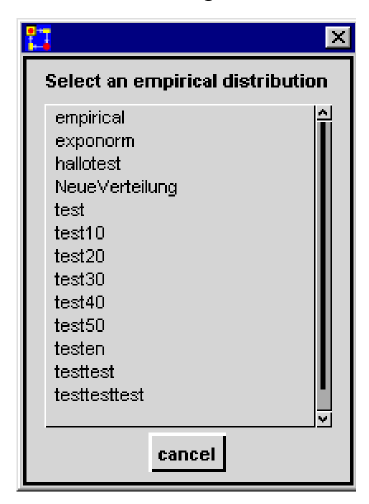


Abb. 5-26: Auswahlfenster für empirische Verteilungen

[7] Die Erzeugung empirischer Verteilungen ist in Kapitel 10 beschrieben.

Selektiert man eine Zeile, so wird die Dichte- oder die Verteilungsfunktion der ausgewählten empirischen Verteilung angezeigt.

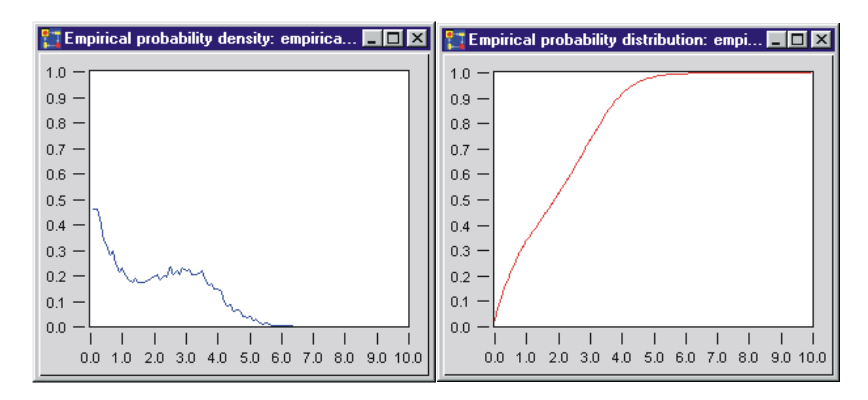


Abb. 5-27: Empirische Wahrscheinlichkeitsdichte und Verteilungsfunktion

5.5.3 SQL-Abfragen

Gelegentlich weicht die Syntax einzelner SQL-Kommandos für verschiedene Datenbanken geringfügig vom Standard ab. Außerdem realisieren verschiedene Datenbanken nur einen Teil des Standards und/oder bieten darüber hinausgehende zusätzliche Kommandos an.

Ist man sich im Einzelfall nicht sicher, so empfiehlt sich ein SQL-Test unter Verwendung des SQL-Query-Tools. Auch einige Datenbanksysteme bieten solche Tests an und erzeugen im Dialog mit dem Anwender sogar das SQL-Kommando für einen bestimmten Datenzugriff (z.B. Microsoft Access).

Im folgenden werden die Schritte für die Ausführung einer SQL-Anweisung für Windows 2000 gezeigt. Die Schritte 1 und 2 brauchen nur einmal durchgeführt zu werden. Die getroffene Auswahl wird in Windows gespeichert und steht auch nach einem Neustart wieder zur Verfügung.

Schritt 1: ODBC-Administrator
Zum Öffnen von Leistungsindikatoren (ODBC) klicken Sie in der Windowsleiste am unteren Rand des Bildschirms auf Start, zeigen auf Einstellungen und klicken dann auf Systemsteuerung. Doppelklicken Sie auf Verwaltung und anschließend auf „Datenquellen (ODBC)“.

Sie können stattdessen auch die Hilfe im Startmenu aufrufen, darin als zu suchendes Schlüsselwort ODBC eingeben, ODBC-Adminstrator auswählen und im Beschreibungsfenster auf „Leistungsindikatoren (ODBC)“ doppelklicken.

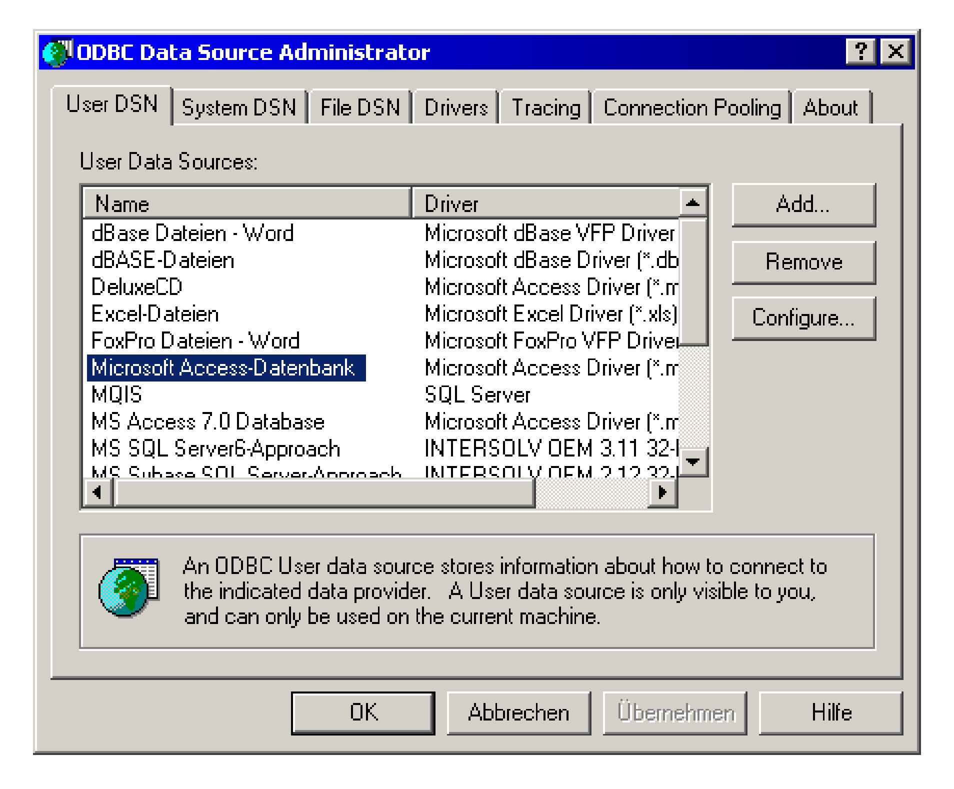


Abb. 5-28:ODBC Data Source Administrator unter Windows 2000

In dem Fenster auf die Zeile doppelklicken, die der anzusprechenden Datenquelle zugeordnet ist. Es geht ein weiteres Fenster auf, das im Schritt 2 besprochen wird. Im vorliegenden Beispiel wurde 'Microsoft Access-Datenbank' ausgewählt.

Schritt 2: ODBC Setup

Abb. 5-29 zeigt das 'ODBC-Setup'-Menu für die Microsoft Access Datenbank. Ersichtlich wurde schon die Datenbank: 'C:\Datenbanktest\DB1.MDB' zugeordnet. Die Zuordnung wird vorgenommen, indem der Knopf 'Auswählen' gedrückt wird (siehe Schritt 2a).

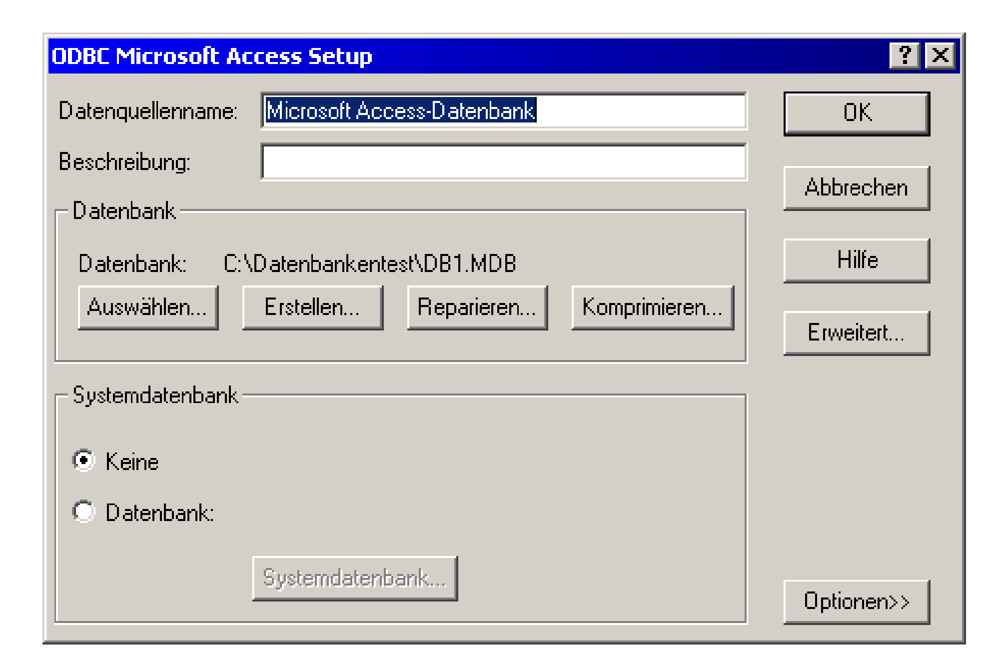


Abb. 5-29: ODBC Setup

Schritt 2a: Auswählen der Datenbank

Die Datenbank wird über das übliche Windows-Auswahlmenü festgelegt.

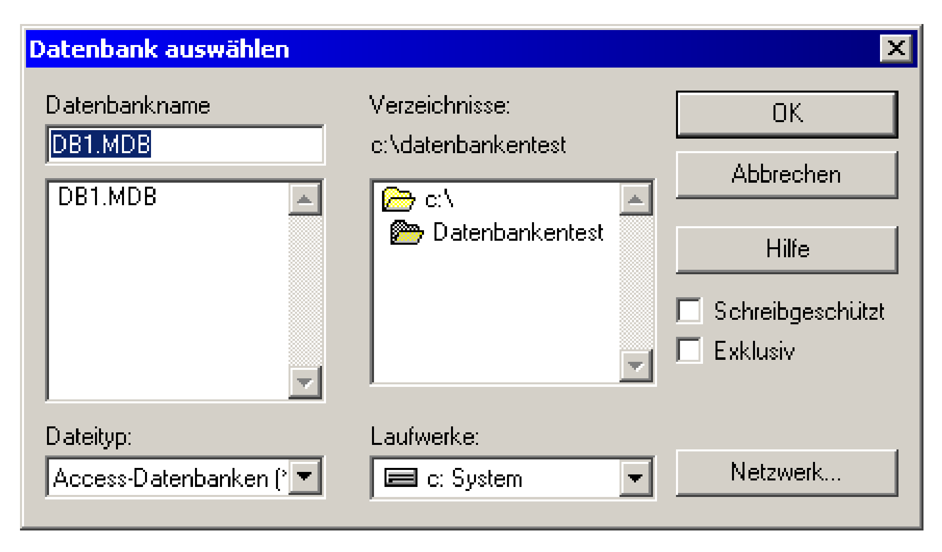


Abb. 5-30: Auswählen der Datenbank

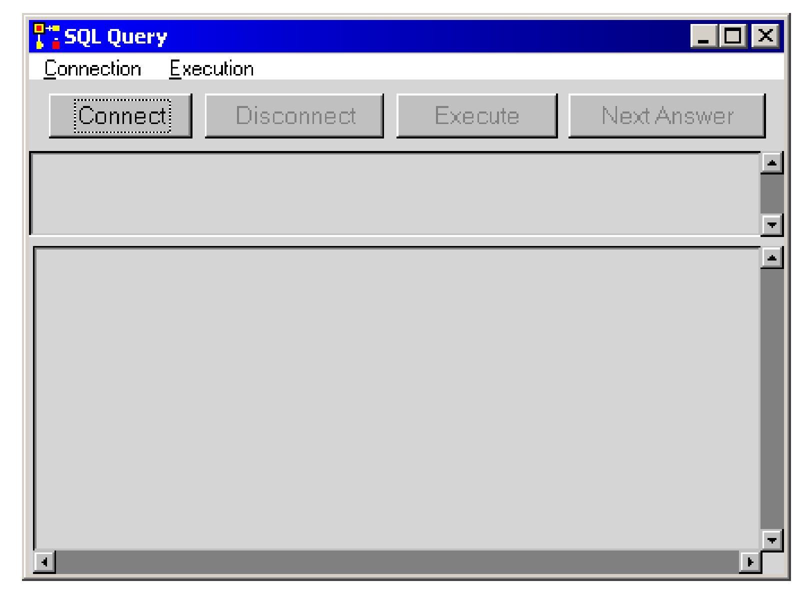


Abb. 5-31: SQL Query aufrufen

Schritt 3: SQL Query Aufrufen
Im PACE-Evaluator-Menü die Funktion 'SQL Query' aufrufen. Es wird das in Abb. 5-31 dargestellte Dialog-Menü angezeigt.

Auf den Knopf 'Connect' drücken, um die Datenbank an das Modell anzuschließen.

Schritt 3a: Datenbank anschließen
Für die vorliegende Datenbank braucht kein Benutzername und kein Passwort eingegeben zu werden. Nur die Umgebung 'Microsoft Access-Datenbank' (siehe die vorhandenen Umgebungen in der Abbildung in Schritt 1) ist einzugeben.



Abb. 5-32: Datenbank anschließen

Nach Drücken von 'ok' wird das Fenster geschlossen und es wird wieder das Fenster aus Schritt 3 angezeigt. Wenn die Datenbank angeschlossen werden konnte, ist jetzt statt dem Knopf 'Connect' nur der Knopf 'Disconnect' wählbar. Andernfalls wird eine Fehlermeldung ausgegeben.

Schritt 4: Ausführen einer SQL-Anweisung
In das Eingabefeld unter der Leiste mit den Druckknöpfen eine SQL-Anweisung eingeben. Sobald etwas eingegeben wurde, kann der Druckknopf 'Execute' bedient werden. Nach vollständiger Eingabe der SQL-Anweisung diesen Knopf drücken.

In dem Ausgabefeld darunter wird das Ergebnis der SQL-Anweisung dargestellt.

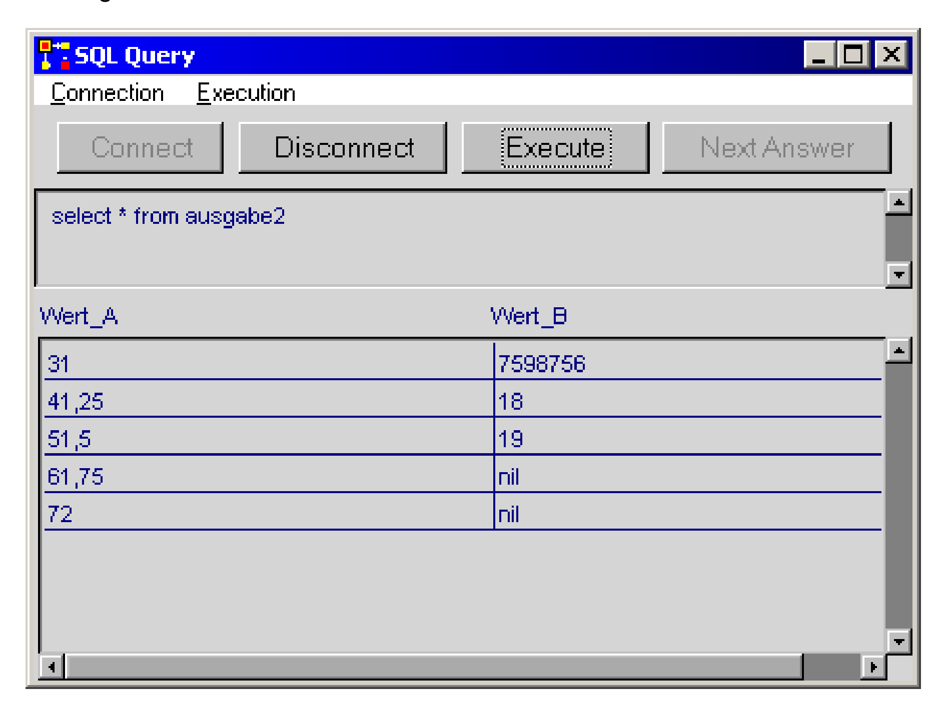


Abb. 5-33: Ausführen einer SQL-Anweisung

Schritt 5: Datenbank abkoppeln
Nach Abschluss aller SQL-Abfragen mit der ausgewählten Datenbank wird die Datenbank wieder freigegeben. Dazu den 'Disconnect'-Knopf drücken.

5.5.4 Fuzzy-Reduktionen

Bei der Bearbeitung von Fuzzy-Problemen sind häufig Vereinfachungen durchzuführen, um die Rechnungen einfacher und schneller zu gestalten.

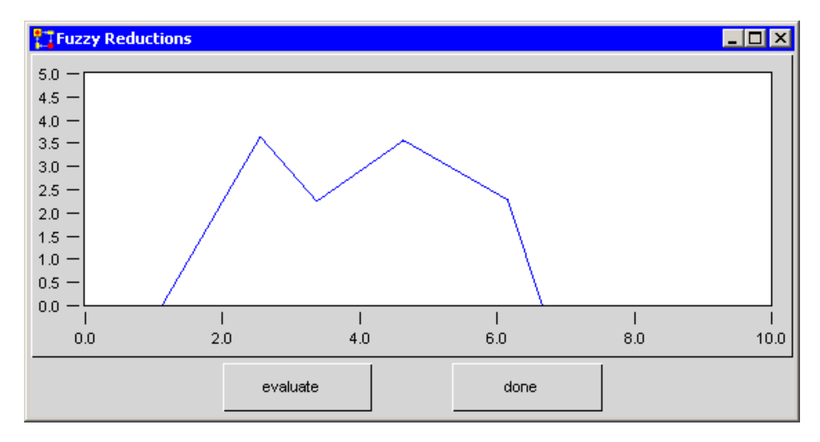


Abb. 5-34: Eingabe-Fenster für Fuzzy-Arrays

Die Menüfunktion 'fuzzy reductions' öffnet das in Abb. 5-34 dargestellte Eingabefenster, in das ein Fuzzy-Array grafisch eingegeben oder ein berechneter Fuzzy-Array eingelesen werden kann.

Bei direkter Eingabe wird mit der rechten Maustaste ein Punkt im Fenster angeklickt, daraufhin der Cursor auf die Position eines weiteren Punktes verschoben und erneut geklickt. PACE verbindet diese Punkte durch eine gerade Linie. Auf diese Weise kann relativ schnell ein Polygonzug gezeichnet werden. Die Eingabe und das Editieren von Kurven ist ausführlich in Abschnitt 6.9.1.4 beschrieben.

Nach der Eingabe des Fuzzy-Arrays wird der Knopf 'evaluate' gedrückt. Daraufhin öffnet sich das in Abb. 5-35 gezeichnete Ergebnisfenster.

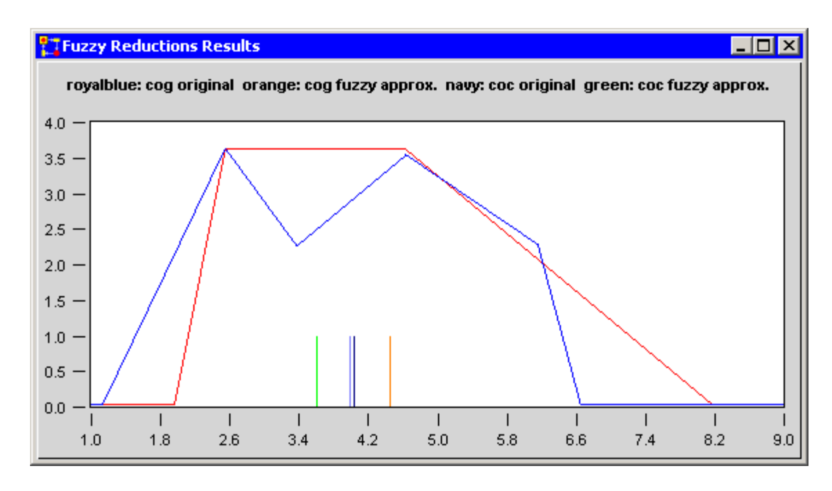


Abb. 5-35: Ergebnisfenster für Fuzzy-Reduktionen

In ihm wird der ursprünglich eingegebene Fuzzy-Array und seine PACE-Trapez-Approximation gezeichnet.[8] Der Anwender kann damit visuell feststellen, ob die in PACE realisierte Trapez-Approximation für seine Aufgabe geeignet ist.

Außer der Originalkurve und ihrer Trapez-Approximation werden für beide Polygonzüge auch die nach der Schwerpunktmethode (cog = center of gravity) und nach der PACE-Eckpunktmethode (coc = center of corners) berechneten scharfen Ausgabewerte (crisp-Werte) durch jeweils zwei senkrechte Linien der Höhe 1 dargestellt. Dabei gelten folgende Farbzuordnungen:

royalblue: cog der Originalkurve
orange: cog der Trapezapproximation
navy: coc der Originalkurve
green: coc der Trapezapproximation

[8] Diese ergibt sich aus dem Fuzzy-Array mit der Wandlungsfunktion asFuzzyTrapezoid (diese Funktion ist im PACE Handbuch über Fuzzy Technik beschrieben).

5.6 Net-Editor-Menü

Über das Netz-Editor-Menü werden die allgemeinen bei der Bearbeitung von Netzen erforderlichen Hilfsmittel zugänglich. Die lokal innerhalb eines Netzfensters erforderlichen graphischen Editierfunktionen sind innerhalb dieses Fensters über die rechte Maustaste erreichbar.

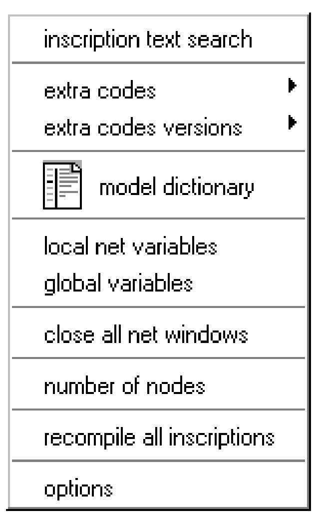


Abb. 5-36: Netz-Editor-Menü

5.6.1 Text-Suche in Inskriptionen

Die Menü-Funktion ‘inscription text search’ öffnet ein Fenster (siehe Abbildung 5-37) in dem alle oder ein Teil der Inskriptionen nach bestimmten Texten abgesucht werden können. Mit Schaltern kann der Suchkontext festgelegt werden.

Mit zwei weiteren Knöpfen 'full search' und 'subtree search' wird festgelegt, ob das gesamte Modell durchsucht oder nur von einem in

der Netzliste selektierten Netz an abwärts gesucht werden soll. Wird einer der beiden Schalter gedrückt, so öffnet sich ein Eingabefenster, in dem der zu suchende Text einzugeben ist. Die Elemente, die den gesuchten Text enthalten, werden in eine Liste eingetragen, die in einem sich öffnenden Fenster (siehe Abbildung 5-38) angezeigt wird. Bei der Suche wird zwischen Groß- und Klein-schreibung unterschieden.

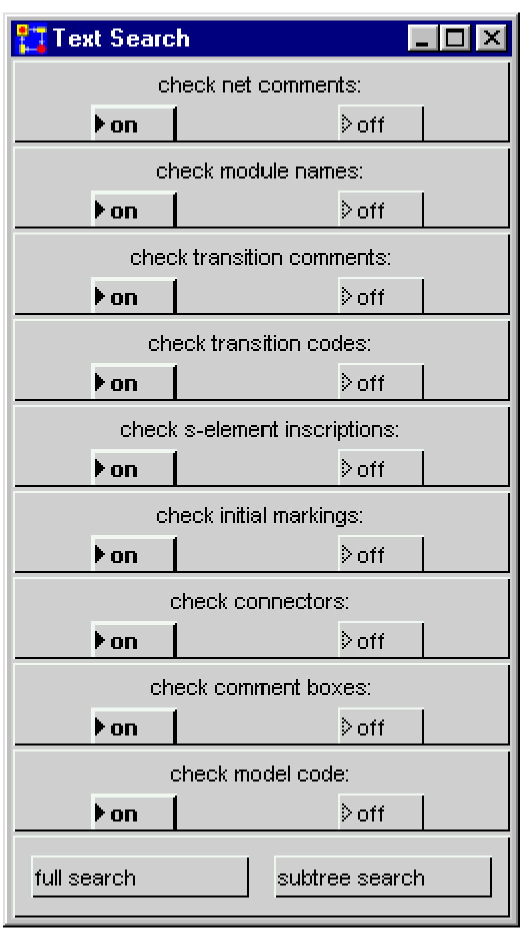


Abb. 5-37: Auswahlfenster für die Textsuche

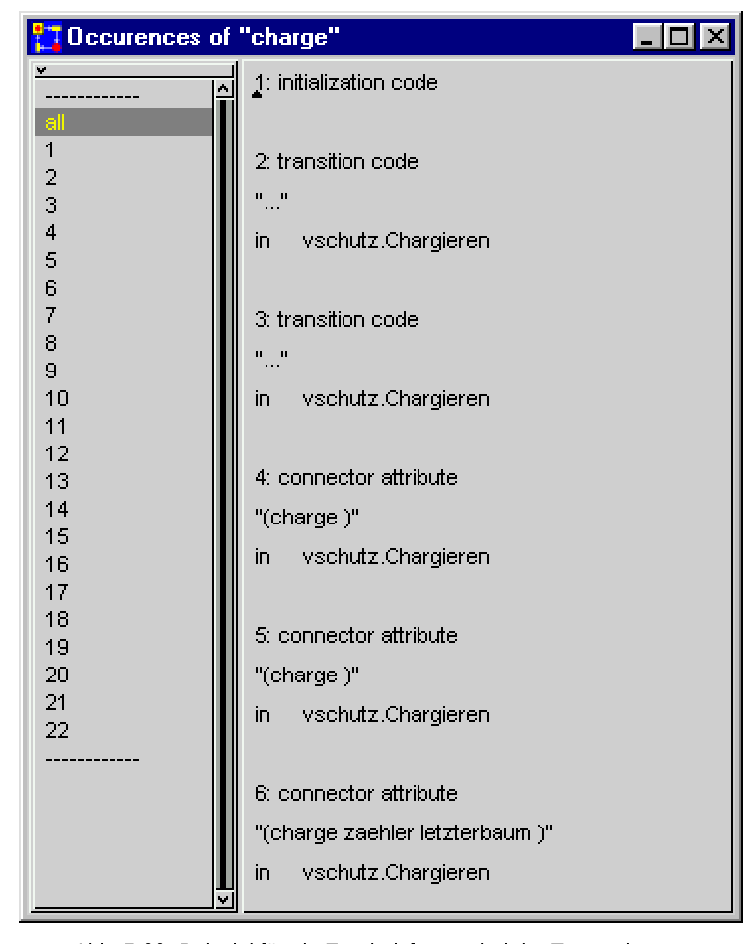


Abb. 5-38: Beispiel für ein Ergebnisfenster bei der Textsuche

Selektiert man eine der Zahlen in der linken Spalte und drückt danach die rechte Maustaste, so wird das folgende einzeilige Menü angezeigt:

open window

Abb. 5-39: Öffnen eines Netzfensters mit dem gesuchten Text

Nach Ausführen des Menüpunkts ‘open window’ öffnet sich das Netzfenster, in dem die Inskription mit dem gesuchten Text auftritt. Schiebt man den Mauszeiger in das Fenster, so wird er automatisch auf dem Netzelement positioniert, das die Inskription mit dem gesuchten Text enthält. Letzteres passiert sofort, wenn das Netzfenster schon geöffnet ist.

5.6.2 Extra-Codes

Durch Ausführen der Funktionen des Submenüs werden jeweils Editierfenster geöffnet, in denen Smalltalk-Code eingegeben wird, der bei Eintritt bestimmter Ereignisse ausgeführt werden soll.

5.6.2.1 Initialization-Code

Der Start- oder Initialization-Code ist Smalltalk-Code, der zu Beginn eines Modell ausgeführt wird. Dieser Code wird beim Anwählen des Befehls 'initialize' (im Simulations-Modus eines Netzfensters), bei Bedienen des Start-Knopfs in einer Exekutive und beim programmgesteuerten Wiederstart eines Modells (restart-Botschaft) ausgeführt. Der Initialisierungs-Code wird benutzt, um die Voreinstellungen für dem Ablauf des Modells durchzuführen, z.B. um Initialwerte globaler Variablen zu setzen, um Files zu öffnen, um externe Geräte zu initialisieren, usw.

5.6.2.2 Break-Code

Der Anhalte- oder Break-Code ist Smalltalk-Code, der beim Anhalten der Simulation (Drücken der linken Maustaste während eines Simulationslaufs) ausgeführt wird.

Der Anhaltecode kann z.B. benutzt werden, um bestimmte Zwischenergebnisse anzuzeigen.

5.6.2.3 Continuation-Code

Der Fortsetzungs- oder Continuation-Code ist Smalltalk-Code, der beim Fortsetzen der Simulation ausgeführt wird.

Eine wichtige Anwendung des Fortsetzungscodes ist die automatische Übernahme von Parametern, die nach dem Anhalten eines Simulationscodes vom Bediener umgesetzt wurden. Auf diese Weise kann der Bediener während der Simulation in die Abläufe eingreifen und das Simulationsgeschehen beeinflussen (Interaktive Simulationsläufe). Er kann z.B. bei Erkennen von Problemsituationen die Simulation anhalten, die zur Behebung der Situation erforderlichen Parameter eingeben und dann die Simulation mit den neuen Parametern fortsetzen. Anwendungsgebiete, bei deren Modellierung Fortsetzungscode eingesetzt werden kann, sind z.B.

- neue technische Systeme, deren Verhalten ggf. in einer vorzugebenden Umgebung per Simulation untersucht werden soll, oder
- logistische Systeme, deren Handhabung eingeübt werden muß (z.B. Trainingssysteme für Disponenten).

5.6.2.4 Termination-Code

Der Beendigungs- oder Termination-Code ist Smalltalk-Code, durch dessen Ausführung der Durchlauf eines Modells abgeschlossen wird. Der Code kann auch im Simulations-Modus durch Anwählen des Befehls 'terminate' oder durch Bedienen des Terminate-Knopfs in einer Exekutive ausgeführt werden. Der Beendigungs-Code wird beispielsweise benutzt, um Files zu schließen, um Ergebnisse eines Simulationslaufs weiterzuverarbeiten oder um diese Ergebnisse auszugeben.

5.6.3 Versionshaltung für Extra-Codes

Gelegentlich ist es nützlich, wenn verschiedene Versionen der Extra-Codes verfügbar sind, sei es, daß man verschiedene Varianten eines Modells in einem PACE-Imagefile halten will oder daß man während der Entwicklung bestimmte Zwischenschritte für eventuelle spätere Verwendung aufheben will

Für diese Zwecke wurde eine einfache Versionshaltung bereitgestellt, die über den Menüpunkt „extra code versions“ angesprochen werden kann. Nach dem Anwählen des Menüpunkts erscheint das Menü:

initialization code versions
break code versions
continuation code versions
termination code versions

Abb. 5-40: Auswahl des Extra-Codes

Darin kann der Extra-Code, dessen Versionshaltung bearbeitet werden soll, ausgewählt werden. Nach der Auswahl erscheint ein Editfenster, für die Versionen des ausgewählten Extracode.

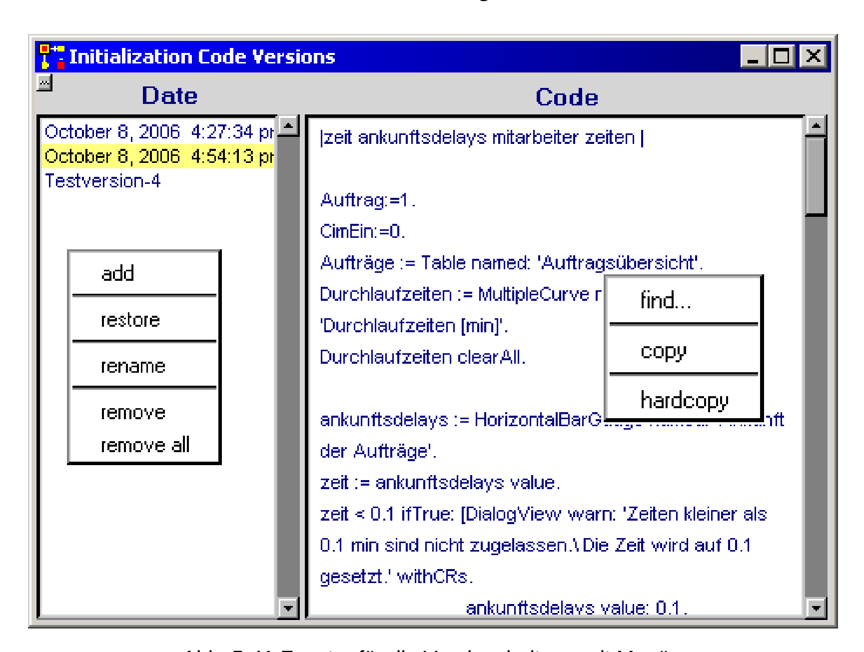


Abb. 5-41:Fenster für die Versionshaltung mit Menüs

Im linken Teilfenster wird defaultmäßig das Datum und die Uhrzeit des Eintrags notiert. Diese Zeitangabe kann vom Anwender durch eine beliebige andere Bezeichnung ersetzt werden. Wird eine Bezeichnung im linken Teilfenster markiert, so wird im rechten Teilfenster der zugehörige Modelcode angezeigt.

Menü im linken Teilfenster:

add Die aktuelle Version des ausgewählten Modelcodes wird als nächster Eintrag in die Versionshaltung aufgenommen. Es öffnet sich ein Abfragefenster für die Bezeichnung der Version. Wird nichts eingegeben, so wird in das linke Fenster das Datum und die Uhrzeit, zu dem dieser Eintrag erfolgt ist, eingetragen.

restore Der aktuelle Modelcode wird mit dem zu dem markierten Eintrag gehörigen Modelcode überschrieben.

rename Der Anwender kann die markierte Bezeichnung durch eine beliebige andere Bezeichnung ersetzen.

remove Die markierte Version wird in der Versionshaltung gelöscht.

remove all Alle gespeicherten Versionen werden in der Versionshaltung gelöscht.

Menü im rechten Teilfenster:

find Es öffnet sich ein Fenster, in dem der zu suchende Text eingegeben wird. Der gefundene Text wird markiert.

copy Der markierte Text wird im Zwischenspeicher gespeichert und kann an anderer Stelle mit dem paste-Kommando wieder eingesetzt werden.

hardcopy Der Text wir auf einen Drucker ausgegeben.

5.6.4 Lokale Netz-Variablen (Modul-Variablen)

Für jeden Modul kann ein Fenster geöffnet werden, in dem Modul-Variable vereinbart, der Wert von Modul-Variablen festgelegt und der aktuelle Wert von Modul-Variablen abgefragt werden kann (Abb. 5-42). Hierzu ist zunächst der gewünscht Modul in der Netz-Liste anzuwählen und danach im 'net editor'-Menü der PACE Haupt-Menüleiste die Funktion 'local variables' auszuführen.

Modul-Variablen-Fenster sind in zwei Teile unterteilt. Links werden die einzelnen Namen der Modul-Variablen in Symbolform (d.h. mit vorangestelltem Nummernzeichen #) aufgelistet. Rechts wird der Code oder der Wert (Initialwert oder aktueller Wert) der jeweils angewählten Modul-Variablen dargestellt.

In dem rechten Fenster kann in zwei verschiedenen Modi gearbeitet werden, die durch Bedienen der Druckknöpfe am unteren Fensterrand eingestellt werden. Im ersten Modus ('initial value') kann ein Anfangswert für eine im linken Fenster selektierte Modulvariable festgelegt werden, der beim Initialisieren eines Modells zugewiesen wird. Im zweiten Modus ('current value') kann der aktuelle Wert der Modul-Variablen angezeigt und/oder geändert werden. Nach der Eingabe eines neuen Werts, ist im rechten Fenster durch Drücken der rechten Maustaste das Standard-Smalltalk-Menü anzuzeigen und darin die Funktion ‘accept’ auszuführen. Im linken Fenster werden mit der Menüfunktion ‘insert’ weitere lokale Netz-Variablen in Symbolform hinzugefügt, mit der Menüfunktion ‘remove’ wieder entfernt.

Abb. 5-42: Fenster für lokale Netz-Variablen

5.6.5 Globale Variablen

Die Ausführung der Funktion ‘global variables’ öffnet ein Fenster, mit zwei Teilfenstern (Abb. 5-43).

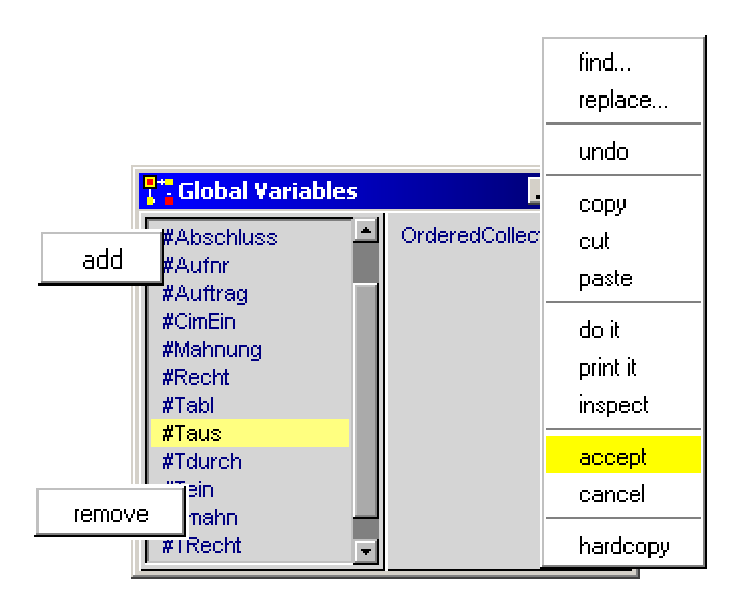


Abb. 5-43: Fenster für globale Variable

Auf der linken Seite werden sämtliche globalen Variablen des Netzes in Symbolform angezeigt. Bei Selektion einer globalen Variable im linken Fenster, zeigt das rechte Fenster den aktuellen Wert der globalen Variablen an. Dieser kann analog zu der Vorgehensweise bei lokalen Variablen geändert werden.

Globale Variable werden entweder über das add-Menü (mit nachfolgendem Eintrag in das rechte Fenster) oder über den Inskriptionscode vereinbart und initialisiert (siehe Abschnitt 3.8.4). Durch Ausführen der ‘remove’-Funktion im linken Teilfenster kann eine selektierte globale Variable aus dem Netz entfernt werden.

5.6.6 Modell-Lexikon

Bei der Bearbeitung von Modellen steht man, insbesondere wenn längere zeitliche Unterbrechungen auftreten, vor dem Problem, daß die Bedeutung der einzelnen Bezeichner vergessen wird oder nur noch oberflächlich bekannt ist. Noch schwieriger ist es, wenn sich neue Mitarbeiter in schon vorhandene Modelle einarbeiten müssen, um diese zu pflegen oder um sie zu erweitern.

In all diesen Fällen ist ein Modell-Lexikon sehr hilfreich. Seine Erstellung kann mit der Erstellung des Modells inkrementell und mit einem Bruchteil des Aufwands vollzogen werden, der bei Fehlen eines solchen Lexikons für das Erarbeiten der Information aus einem vorliegenden, schlecht dokumentierten Modell erforderlich ist.

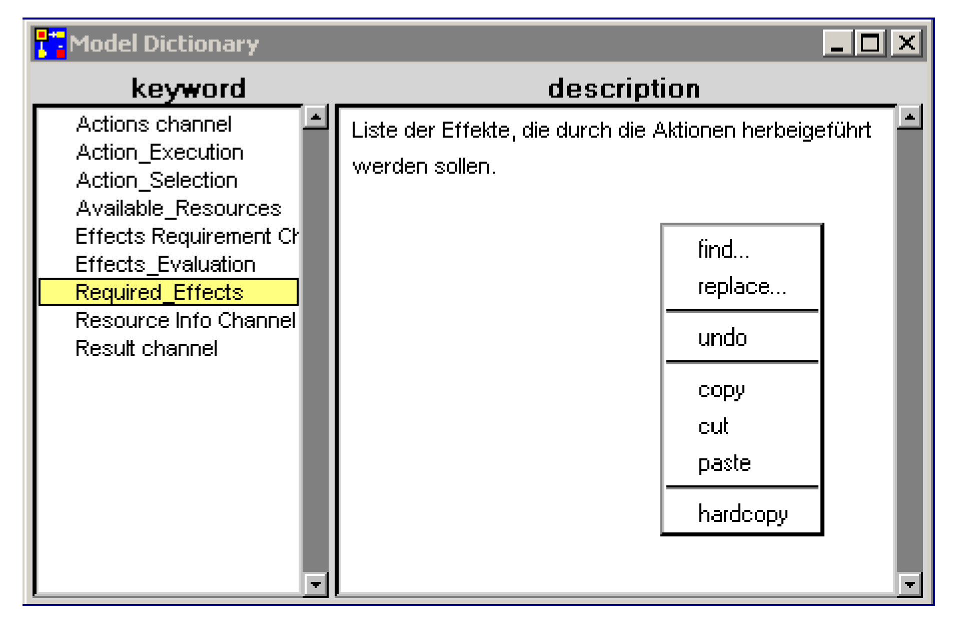


Abb. 5-44:Modell-Lexikon

Neben der Möglichkeit, Texte direkt in das Lexikon einzugeben, bietet PACE in allen Menüs für die Bearbeitung von Inskriptionen eine zusätzliche Funktion 'dictionary' an, mit der markierte Textstücke automatisch als Entry in das Model Dictionary aufgenommen werden können. Bei Auswahl der Funktion wird das Dictionary mit dem neuen Entry angezeigt und man kann dann sofort zugeordnet den erklärenden Text eingeben.

Im linken Teilfenster kann mit der rechten Maustaste das folgende Menü angezeigt werden:

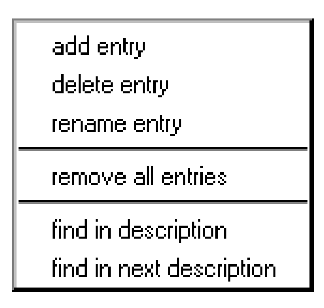


Abb. 5-45: keyword-Menü

Die Menü-Funktionen haben folgende Bedeutung:

add entry
In das sich öffnenden Fenster kann ein Text eingegeben werden, der als weiterer Entry in das Dictionary eingetragen wird.

delete entry
Entfernt den markierten Entry aus dem Dictionary.

rename entry
Es öffnet sich ein Fenster, in dem der geänderte Entry-Text eingegeben werden kann.

remove all entries
Erzeugt ein leeres Dictionary durch Löschen aller Einträge im Dictionary.

find in description
Es öffnet sich ein Fenster, in das der Text, nach dem gesucht werden soll, einzugeben ist. Nach Eingabe des Textes wird die Suche durch Drücken der Return-Taste (Tastatur) oder durch Drücken des OK-Knopfs (Maus) gestartet.

Die Suche bearbeitet die den Keywords zugeordneten Texte ausgehend von dem aktuell markierten Keyword und hält an, sobald ein Text gefunden wurde, der mit dem Suchtext übereinstimmt, oder wenn alle Texte durchsucht worden sind. Falls die Suche erfolgreich ist, wird die Beschreibung angezeigt, in der der fragliche Text gefunden wurde. Der gefundene Text wird markiert.

find in next description
Ausgehend von dem mit 'find in description' gefunden Text werden die Texte weiter nach dem dort angegebenen Suchtext durchsucht. Der Suchtext braucht also nicht neu eingegeben zu werden.

Ansonsten ist das Verhalten identisch mit 'find in description'.

5.6.7 Schließen aller Netzfenster

Mit der Funktion ‘close all net windows’ werden alle über die Funktionen des Netz-Listenfensters direkt oder indirekt geöffneten Netzfenster geschlossen. Das Netz-Listenfenster kann nur durch Ausführen der Funktion ‘leave net’ im ‘file’-Menüs geschlossen werden.

5.6.8 Anzahl der Netzknoten

Die Funktion ‘number of nodes’ öffnet ein Fenster und zeigt darin die Anzahl der Knoten eines Netzes an. Netzknoten sind alle Stellen und Transitionen eines Netzes.

5.6.9 Neu-Übersetzen aller Inskriptionen eines Netzes

Durch Ausführen der Funktion 'recompile all inscriptions' wird die Neu-Übersetzung aller Inskriptionen eines Netzes zwar vorbereitet,

aber noch nicht durchgeführt. Die Durchführung erfolgt automatisch bei der nächsten Initialisierung des Netzes.

Die Inskriptionen der Netz-Elemente werden bei der Entwicklung eines Netzes durch die Standard-Smalltalk-’accept’-Funktion in Code für die virtuelle Smalltalk-Maschine umgesetzt. Die Funktion 'recompile all inscriptions' löscht die interne Form der Inskriptionen aller Elemente des geladenen Netzes. Beim Start des nächsten Simulationslaufs werden danach alle Inskriptionen automatisch neu übersetzt.

Das Ausführen der Menüfunktion kann sich lohnen, wenn ein großes PACE-Image abgespeichert werden soll, weil das neu übersetzte Image normalerweise weniger Speicher benötigt. Die Neu-Übersetzung von Inskriptionen wird aber relativ selten benötigt. Sie wird automatisch beim ersten Initialisieren eines Netzes nach dem Laden des .net-Files durchgeführt.

5.6.10 Net-Editor-Optionen

Die beim Editieren von Netzen einstellbaren Optionen sind in Abb. 5-46 dargestellt. Sie sind für das gesamte gerade in Arbeit befindliche Modell gültig, Die einzelnen Parameter bestimmen das Layout der statischen Netzelemente und werden zusammen mit dem Modell geladen und gespeichert.

Im einzelnen setzen die Optionen folgende Netzparameter:

element size
Definiert die Größe der Netzelemente.

arrow length
Definiert die Länge der Pfeilspitzen der Konnektoren.

arrow angle
Definiert die Schenkelwinkel der Pfeilspitzen der Konnektoren.

inhibitor radius
Definiert die Größe des Inhibitor-Symbols (gefüllter Kreis auf der Connector-Spitze).

grid
Definiert den Abstand der Rasterlinien im Netzfenster, welche die Positionierung der einzelnen Netz-Elemente bestimmt.

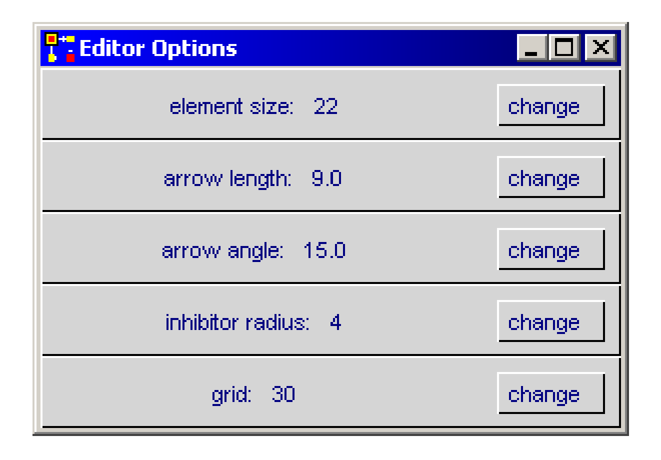


Abb. 5-46: Net-Editor-Optionen-Fenster

5.7 Simulator-Menü

Das Simulator-Menü enthält die allgemeinen Unterstützungsfunktionen für die Ausführung von Netzen. Die lokal innerhalb eines Netzfensters erforderlichen Steuerfunktionen sind innerhalb des Netzfensters über die rechte Maustaste erreichbar.

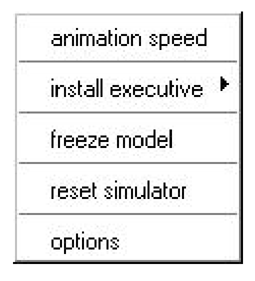


Abb. 5-47: Simulator-Menü

5.7.1 Animationsgeschwindigkeit

Über die Animationsgeschwindigkeit wird festgelegt, wie schnell sich die Marken in einem PACE-Netz bewegen. Sie wird über einen horizontalen Balken eingestellt (Abb. 5-48).

Die Farben des Balken werden über ein Auswahlmenü, daß sich nach Drücken der recht Maustaste öffnet, festgelegt (siehe auch Kapitel 6).

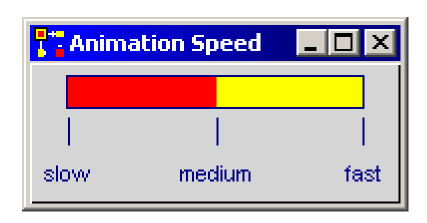


Abb. 5-48: Einstellen der Animationsgeschwindgkeit

5.7.2 Executiven und Einfrieren von Modellen

Es gibt insgesamt acht Spezialleisten (Exekutiven) für die Ausführung von Netzen, vier horizontale und vier vertikale Leisten. Diese sind in jeweils zwei sog. kurze und zwei sog. lange Leisten unterteilt. Sie werden durch Ausführen der Submenü-Funktionen von ‘install executive’ geöffnet.

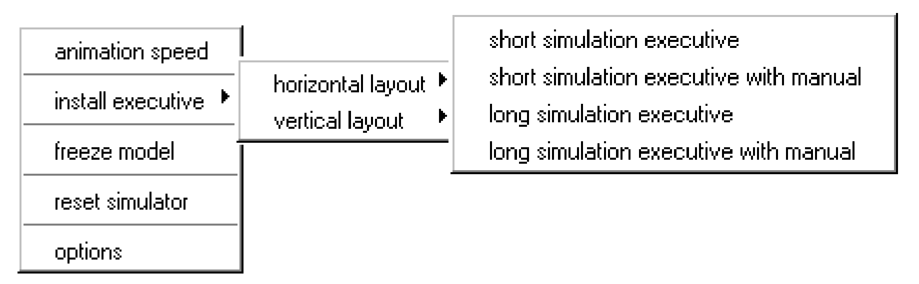


Abb. 5-49: Bereitstellen einer Exekutive

5.7.2.1 Warum braucht man Executiven?

Über die PACE-Hauptleiste und die über sie erreichbaren Menüs können alle Möglichkeiten von PACE ausgeschöpft werden. Die Kenntnis aller PACE-Menüs ist für die Modellentwicklung unabdingbar und erfordert normalerweise eine angemessene Schulung.

Bei vielen Anwendungen von PACE werden jedoch nur fertige Modelle ausgeführt, die vom Anwender nicht verändert werden können. Beispiele hierfür sind

- Monitoring und Darstellung von Verwaltungs- und Produktionsprozessen (z.B. Darstellung von Teilprozessen in verteilten Fertigungssystemen).
- Voraussage von charakteristischen Daten für eine Produktionsanlage, die für die weitere Online-Disposition benötigt werden (z.B. Auslastung von Anlagenteilen bei diskontinuierlichen fremdgesteuerten Zuführungen).
- Kostengünstige Schulungen (z.B. Auftragsbearbeitung oder Disposition, Verteilung und Auslieferung von Gütern).

- Auslegung von Systemen für den elektronischen Handel (e-commerce).

Um für solche Anwendungen den Lernaufwand für PACE auf ein Minimum zu reduzieren, wurden in PACE Exekutiven vorgesehen, über die der Ablauf von Modellen ohne vorhergehende PACE-Schulung gesteuert werden kann. Unter Verwendung der weiter unten beschriebenen Vorgehensweise kann einem Modell eine solche, einfach zu bedienende spezielle Anwendungs-Schnittstelle hinzugefügt werden.

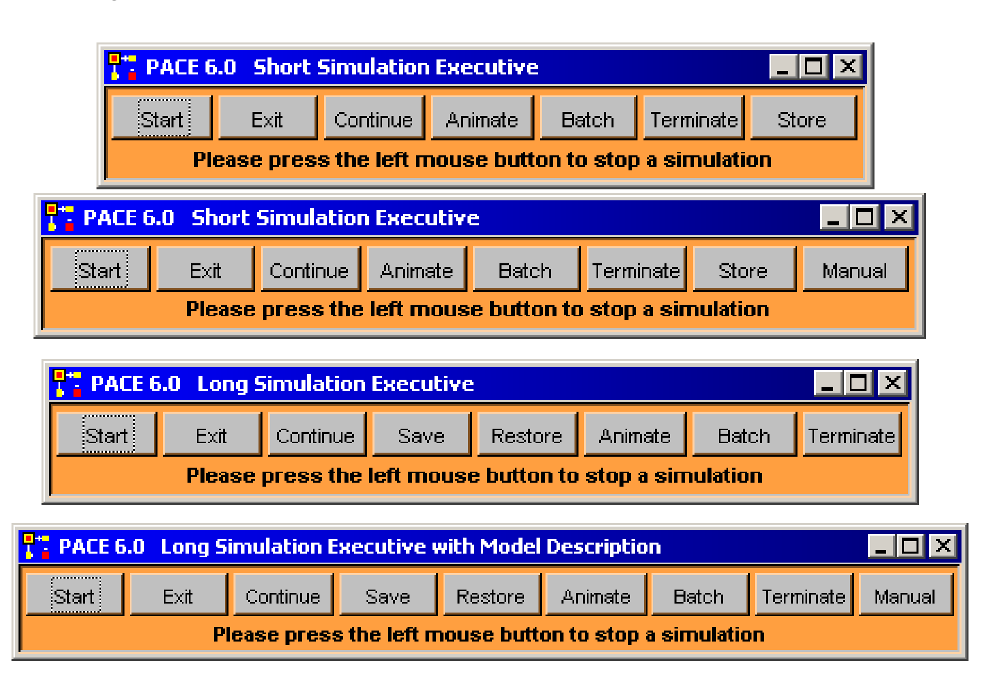


Abb. 5-50: Horizontale Executiven

Durch die Ausführung einer weiteren Funktion kann das Modell mitsamt der Anwendungs-Schnittstelle "vergossen" oder "eingefroren" werden. Danach ist das Modell nur noch über die Bedienleiste und ggf. vorgesehene textuelle oder graphische Eingabefenster steuerbar; alle weiteren geöffneten Fenster, über die das Modell

während seiner Entwicklung bearbeitet werden konnte (insbesondere die Netzfenster), sind nicht mehr bedienbar, d.h. können vom Bediener nicht mehr verändert werden.

Das Vergießen von Modellen hat mehrere Vorteile:

- Fehlbedienungen, durch welche das auszuführende Modell und die ausgewählten Fenster unzulässig verändert würden, sind nicht mehr möglich.

- Bei der Weitergabe von Modellen muß vom Hersteller eines Modells nicht mehr die gesamte Entwicklungs-Umgebung mitgeliefert werden.

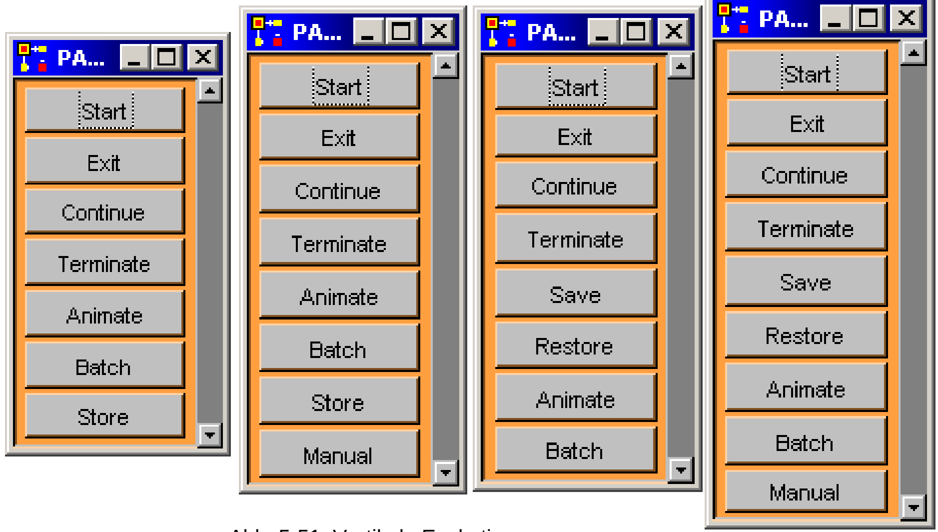


Abb. 5-51: Vertikale Exekutiven

Die Weitergabe vergossener Modelle erfordert eine spezielle PACE-Weitergabe-Lizenz. Vergossene Modelle werden mit den Dateien im bin-Verzeichnis des PACE-Verzeichnisses ausgeliefert.

Die hier beschriebene Vorgehensweise entspricht völlig der Vorgehensweise, die man bei der Entwicklung von Objekt-Programmen unter Verwendung von Programmiersprachen wie FORTRAN oder C

einschlägt. Bei Verwendung dieser Sprachen wird ein komplexer Übersetzungs- und Binde-Prozeß zur Entwicklung von Objekt-Programmen eingesetzt. Nur die Objekt-Programme selbst werden später weitergegeben und können vom Anwender nicht mehr verändert werden.

5.7.2.1 Erstellen einer Anwendung

Die ausführbare Form eines fertig entwickelten PACE-Modells mit einer einfachen Druckknopf-Leiste (im folgenden als 'Anwendung' bezeichnet) kann schrittweise nach der folgenden Liste erstellt werden:

1. Laden der Netzliste des Modells

2. Laden eines Transcript-Fensters (‘work’-Menü der PACE-Hauptleiste). Das Transcript-Fenster in die jeweils gewünschte Bildschirmposition bringen.

3. Den Knopf 'simulator' in der PACE-Hauptleiste drücken und die gewünschte Menüfunktion zum Laden einer Executiven ausführen. Eine Leiste mit Druckknöpfen erscheint und ist an die jeweils gewünschte Bildschirm-Position zu schieben.

4. Alle Fenster mit Netzen, Unternetzen, Diagrammen, Statistik-Fenstern usw., die dem Anwender während der Modellausführung gezeigt werden sollen, laden und auf dem Bildschirm positionieren. Dabei ggf. Szenen definieren und das Auswahlfenster für Szenen laden und positionieren. Mindestens ein Netzfenster muss im Simuationsmodus geöffnet werden.

5. Ausprobieren des Modells unter Verwendung der Knopf-Leiste und ggf. noch verändern, bis der Ablauf und das Aussehen des Modells befriedigt.

   Das Modell kann in diesem Zustand mit der Funktion 'store image' im File-Menü für spätere Änderungen abgespeichert werden. Der Image-Name sollte von dem des Entwicklungs-

Images, von dem unter Punkt 1. ausgegangen worden ist, verschieden sein!

6. Erneut den Knopf 'simulator' in der Hauptleiste drücken und die Funktion 'freeze model' ausführen. Die PACE-Hauptleiste und das Netzlistenfenster verschwinden. Alle Netzfenster werden in den Simulationsmodus gebracht. Die übrigen Fenster bleiben unverändert. Die PACE-Funktionen aller Fenster außer dem Fenster mit der Knopf-Leiste, den Graphik-Fenstern und Tabellen sind danach nicht mehr bedienbar.

   Es erscheint ein Fenster, in dem der Name des Images anzugeben ist, unter dem das 'eingefrorene' Modell abgespeichert werden soll. Dieser Name sollte von dem des Entwicklungs-Image und dem unter Punkt 5. gespeicherten Sicherungs-Image verschieden sein! Deshalb den Namen des Anwendungs-Images sorgfältig auswählen!

   Nach Eingabe des Namens wird das Modell abgespeichert und beendet.

Sie können nun die Anwendung in der üblichen Weise über ein mit den Windows-Funktionen einzurichtendes Ikon, über den Explorer oder über die Kommando-Schnittstelle starten.

5.7.2.2 Funktionen der Druckknöpfe

Wie Abb. 5-50 und Abb. 5-51 zeigen, sind derzeit vier horizontale und vier vertikale Exekutiven vorgesehen, die jeweils in zwei kurze und zwei lange Exekutiven unterteilt sind und sich durch eine unterschiedliche Anzahl von Knöpfen und Funktionen unterscheiden.

Die kurzen Exekutiven dienen vorzugsweise für die reine Modellausführung. Die langen Executiven werden unter anderem im Schulbetrieb bei der Ausführung von Trainingsmodellen verwendet, wie sie z.B. bei der Ausbildung von Disponenten eingesetzt werden. Sie ermöglichen die Rückkehr zum letzten Save-Punkt, wenn bei der

weiteren Arbeit mit dem Modell das angestrebte Ziel nicht erreicht wurde.

Die Funktionen der Druckknöpfe aller Leisten werden im folgenden gemeinsam beschrieben.

Start
Initialisiert das geladene Modell und startet die Modellausführung.

Exit
Beenden der Modellausführung. Vor dem Beenden kann der Anwender je nach den Funktionen des Leiste den aktuellen Stand des Modells wahlweise

- entweder mit der Save-Funktion in das nächste Image 'SAVExx.IM' retten oder
- das Modell mit der store-Funktion in eine beliebige Datei abspeichern.

Continue
Setzt die Modellausführung nach einer Unterbrechung fort.

Terminate
Diese Funktion führt den ‘termination code’ des Models aus.

Save
Nach einer Unterbrechung des geladenen Modells kann der aktuelle Netz-Zustand des Modells in einem Image 'SAVExx.IM' im Arbeitsverzeichnis gespeichert werden. xx nimmt bei der ersten Sicherung den Wert 01 an. Bei weiteren Sicherungen wird der Wert xx jeweils um 1 erhöht (02 03 usw.).[9] Es sind maximal 99 Sicherungen möglich.

Im Schulbetrieb kann die Funktion dazu verwendet werden, um dem Lehrer die Möglichkeit zu geben, anhand von Zwischenständen die Fortschritte von Schülern zu beurteilen. Zusammen mit der weiter unten beschriebenen Funktion 'Restore' kann die Funktion im Schulbetrieb für das Einüben von organisatorischen Vorgehensweisen verwendet werden.

---

[9] Der Simulator legt beim Start über eine lange Leiste automatisch ein Image save00.imm an, das eine Kopie des Ausgangs-Images ist.

Restore
Durch Drücken des Restore-Knopfs wird der aktuelle geladene Modellzustand verlassen und zu dem Modellzustand zurückgekehrt, der beim vorangegangenen Sichern vorlag.

Animation
Durch Drücken dieses Knopfs vor dem Starten oder Fortsetzen einer Modelausführung wird der Animator für die Modellausführung eingeschaltet. Falls das Einstellfenster für die Markengeschwindigkeit (Funktion ‘animation speed’ im Simulator-Menü) angezeigt wird, kann der Anwender des Modells die Geschwindkeit der Animation verändern.

Batch
Durch Drücken dieses Knopf vor dem Starten oder Fortsetzen eines Simulationslaufs, wird bestimmt, dass das Modell im Hintergrund-Modus (ohne Animation) mit der maximal möglichen Geschwindkeit ausgeführt werden soll.

Store
Damit kann der aktuelle Stand eines Modells in eine beliebige Datei abgespeichert werden.

Die Funktion ist in industriellen Anwendungen u.a. nützlich, um das Fehlverhalten von Modellen oder um Ausnahmezustände aufzuzeichnen und für die spätere Analyse festzuhalten.

Manual
Falls während der Modellerstellung mit dem im Support-Menü vorgesehenen Menüpunkt 'model user manual' ein Bedienungsmanual für das Modell erstellt wurde, kann dieses durch Drücken des Knopf 'Manual' angezeigt werden.

Während der Modellausführung (d.h. während eines Simulationslaufs) ist der Cursor innerhalb von PACE-Fenstern nicht sichtbar. Um den Ablauf eines Modells zu unterbrechen, wird der Cursor in eines der PACE-Fenster geschoben und danach die linke Maustaste gedrückt. Danach wird der Cursor wieder in dem Fenster sichtbar, in

dem die letzte Aktion vor der Unterbrechung stattfand.

5.7.3 Zurücksetzen des Simulators

Die Menüfunktion ‘reset simulator’ wird für die Arbeit mit PACE nicht wirklich benötigt. Sie setzt die gesamte Datenstruktur, die vom Simulator aufgebaut wurde, wieder zurück. Der 'reset'-Befehl kann vor dem Abspeichern eines umfangreichen Images sinnvoll sein, um Speicherplatz auf dem Datenträger zu sparen.

5.7.4 Simulator-Optionen

Die für das Ausführen von Netzen einstellbaren Optionen sind in Abb. 5-52 dargestellt. Sie sind für das gesamte Modell gültig, Die einzelnen Angaben bestimmen die während des Ablaufs eines Modells auszuwertenden Parameter. Diese werden zusammen mit dem Modell geladen und gespeichert.

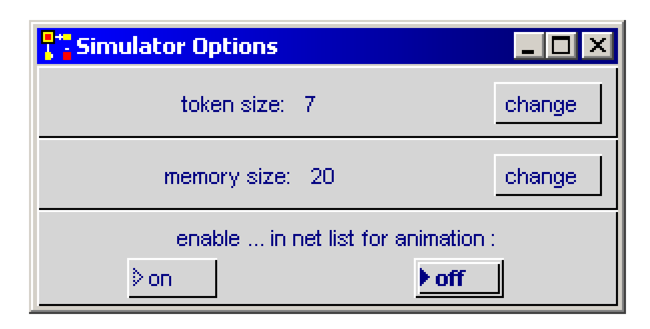


Abb. 5-52: Simulator-Optionen-Fenster

token size
Definiert die Größe der Standard-Ikone für Marken (gefüllter Kreis).

memory size
Definiert die Anzahl der gefeuerten Transitionen, die bei der Simulation mit dem 'back'-Befehl zurückgefahren werden können.

enable ... for animation
Der '...'-Druckknopf (drei Punkte) hat eine Schalterfunktion. Einmal ausgeführt, verhindert er während der Simulation innerhalb eines nicht gebundenen Fensters, daß der Fensterinhalt gegen ein mit dem Drei-Punkte-Befehl '...' gekennzeichnetes Netz und seine Unternetze ausgetauscht wird (siehe auch Abschnitt 5.2.2.2 load net). Durch ein erneutes Betätigen des Druckknopfs wird die Darstellung der '...'--Teilnetze während der Simulation wieder möglich.

5.8 Debugger-Menü

In PACE gibt es zahlreiche Hilfen für das Austesten und Verifizieren von Netzen. Hierzu gehört auch die schon in Abschnitt 5.6.1 besprochene Funktion ‘inscription text search’, die sowohl beim Editieren als auch beim Austesten von Netzen nützlich ist.

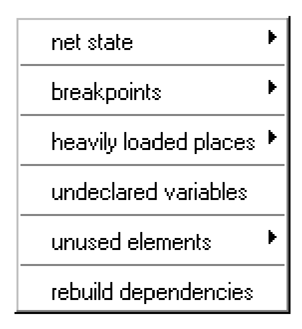


Abb. 5-53: Debugger-Menü

5.8.1 Netzzustand speichern und laden

store state
Der aktuelle Zustand des gesamten Netzes wird in ein File gespeichert, das im Unterverzeichnis 'states' des PACE-Verzeichnisses abgelegt wird.

load state
Ein Zustand des gesamten Netzes, der zuvor abgespeichert wurde, wird wieder geladen.

Bei diesem Befehl muß beachtet werden, daß PACE nicht überprüfen kann, ob die Netzstruktur des Modells zwischen dem Speichern und dem erneuten Laden des Zustandes geändert bzw. ausgetauscht wurde. Der Benutzer muß selbst dafür Sorge tragen, daß der zuvor gespeicherte Simulationszustand wieder in das gleiche Netz geladen wird.

load module state
Lädt aus einem abgespeicherten Zustand des gesamten Netzes den Anteil für das in der Netzliste selektierte Teilnetz.

5.8.2 Unterbrechungspunkte

In PACE gibt es sowohl örtliche als auch zeitliche Unterbrechungspunkte.

node breakpoints
Öffnet ein Fenster mit einer Auflistung aller an Elemente gebundenen Unterbrechungen (näheres hierzu in Abschnitt 8.7.1: 'node breakpoints').

time breakpoints
Öffnet ein Fenster mit einer Auflistung aller zeitlich bedingten Unterbrechungen (näheres hierzu in Abschnitt 8.7.2: 'time breakpoints').

5.8.3 Stark belegte Stellen (heavily loaded places)

Während der Entwicklung von größeren Modellen kommt es gelegentlich vor, dass sich in einzelnen Stellen ungeplant zahlreiche Token ansammeln. In der Regel deutet dies auf einen Modelfehler hin. Er wird dadurch erkennbar, dass das Modell wegen der ausgedehnten Token-Verwaltung der stark belasteten Stellen merkbar langsamer läuft.

Um die stark belegten Stellen zu finden, ist eine Online-Überwachung der Modellausführung oder ein Offline-Analyse des gesamten Modells vorgesehen.

5.8.3.1 Dynamischer Test auf Überladung

Der dynamische Test wird durch den Schalter in dem nachfolgend dargestellten Fenster ein- und ausgeschaltet. Für den Test ist die kritische Anzahl von Marken anzugeben, ab der eine Stelle in dem jeweiligen Modell als überladen angesehen werden soll.

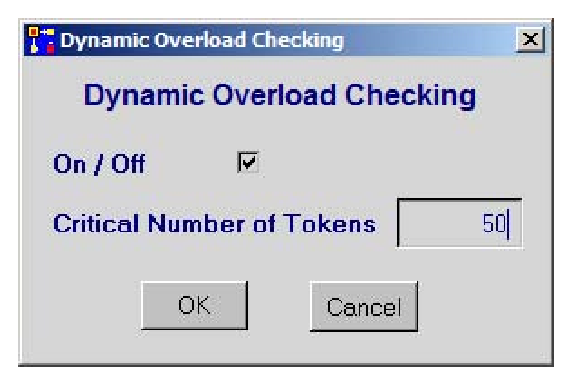


Abb. 5-54: Dynamische Prüfung auf das Überladen einer Stelle

Ist der Test eingeschaltet, so wird der Modellablauf unterbrochen, sobald eine Stelle die kritische Anzahl von Token erreicht. Es öffnet sich dann ein Mitteilungsfenster, in dem der Modulname des Moduls angegeben ist, in dem die fragliche Stelle überladen wurde. Drücken man in diesem Fenster den Knopfs 'Show Place' , so wird das Netzfenster angezeigt, in dem die Stelle definiert ist. Schiebt man nach dem Öffnen des Netzfensters den Cursor in dieses Fenster, so wird er automatisch auf der überladenen Stelle platziert.

Zu beachten ist, dass der Modul, in dem die Stelle definiert ist, nicht mit dem Modul übereinstimmen muss, in dem die Überladung stattgefunden hat. Im Rahmen der hierarchischen Organisation kann eine Stelle in mehreren Modulen auftreten.

5.8.3.2 Modellanalyse auf Überladung

Durch Drücken des 'Find'-Knopfs (siehe die nachfolgende Abbildung) werden sämliche Stellen des Modells daraufhin untersucht, ob sie die angegebene Anzahl von Token enthalten oder überschreiten. Das Ergebnis wird als Liste in dem linken Teilfenster eines Ergebnisfensters angezeigt.



Abb. 5-55: Auffinden aller überladener Stellen

Bei Auswahl eines Listenelements, zeigt das rechte Teilfenster des Ergebnisfensters Information über die ausgewählte Stelle an. Mit der rechten Maustaste kann dann über den Menüpunkt 'open window' das Netzfenster geöffnet werden, in dem Stelle vereinbart ist. Schiebt man nach dem Öffnen des Netzfensters den Cursor in das Fenster, so wird er automatisch auf der überladenen Stelle platziert.

5.8.4 Nicht vereinbarte Variablen

Die Menüfunktion ‘undeclared variables’ zeigt eine Liste der nicht vereinbarten ('undeclared') Variablen in dem aktuellen Modell an. Diese sollte bei normalem Arbeiten mit PACE leer sein.

5.8.5 Nicht verwendete Konnektoren und Marken

unused connectors

Diese Funktion zeigt eine Liste der Konnektoren, die keinen Einfluß auf das Verhalten des Modells haben. Dies sind Konnektoren, die nie eine Transition aktivieren oder inhibieren können..

In dem in Abb. 5-56 dargestellten Beispiel enthält die Stelle ‘Stelle’ nie eine Marke. Deshalb ist der Konnektor überflüssig.

Beispiel:

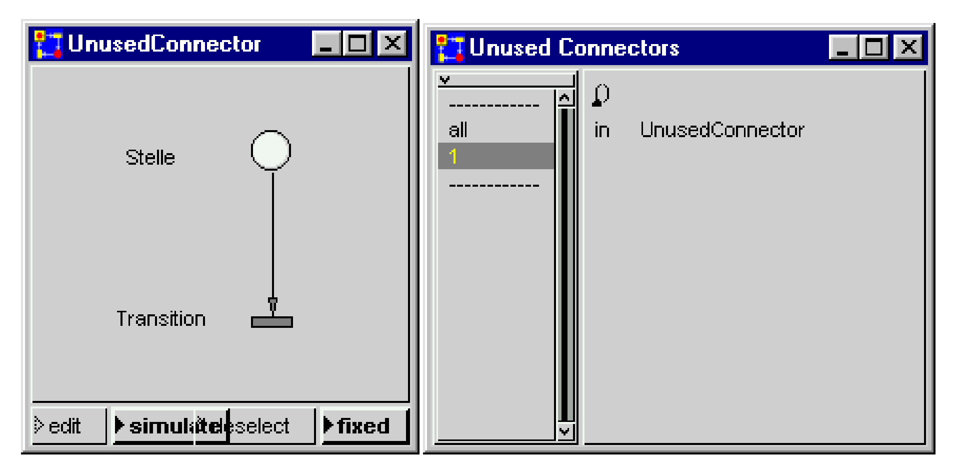


Abb. 5-56: Beispiel für einen überflüssigen Konnektor

unused tokens
Diese Funktion zeigt eine Liste der nicht benötigten Initial-Marken. Eine Marke ist nicht benutzt, wenn sie nie von einer Transition verarbeitet werden kann oder nie eine Transition inhibiert.

5.8.6 Neuaufbau des Datenmodells

Die Funktion ‘rebuild dependencies’ baut die interne Darstellung der Netzstruktur neu auf.

Im Normalfall ist dies nicht nötig. Diese Funktion wird verwendet, wenn das Simulationsverhalten eines Modelles unlogisch erscheint und vermutet wird, das durch die Ausführung von Inskiptionen Daten überschrieben worden sind.

5.9 Extras-Menü

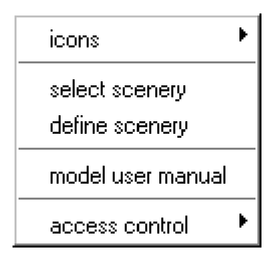


Abb. 5-57: Support-Menü

5.9.1 Ikonen

Wie in der Einleitung schon erwähnt, spielen graphische Elemente, durch die Funktionen eines Netzes unmittelbar transparent werden, bei der Akzeptanz von Netzen eine große Rolle. Über die Funktionen des Ikonen-Menüs und der nachgeordneten Fenster können Bilder (Ikonen, Bitmaps) in PACE übernommen und bearbeitet werden, bevor sie für Komponenten des Netzes oder als Hintergrundbilder eingesetzt werden (siehe dazu Kapitel 7).

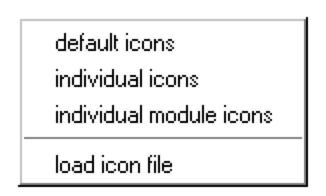


Abb. 5-58: Ikonen-Menü

5.9.1.1 Vorbesetzte Ikonen

Die Menü-Funktion ‘default icons’ öffnet ein Fenster mit dem Default-Ikonen-Satz für das Model, das sich gerade in Arbeit befindet. In diesem Fenster können sowohl die Standard- als auch die

Alternativ-Ikonen neu definiert, geändert oder durch Grafiken ersetzt werden.

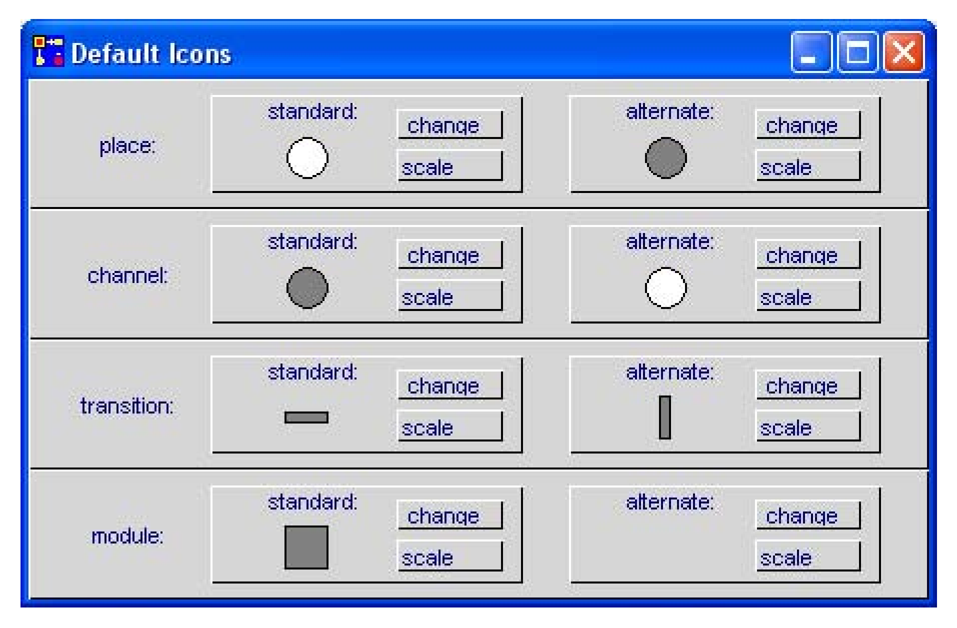


Abb. 5-59:Fenster zum Ändern und Skalieren von vorbesetzten Ikonen



Abb. 5-60:Fenster, das durch Drücken des ‘change’-Knopfs geöffnet wird

Durch Drücken des ‘**change**’-Knopfs wird das in Abb. 5-60 dargestellte Auswahlfenster geöffnet. Bei Auswahl das Zeile ‘individual’ öffnet sich das im nächsten Abschnitt beschriebene Fenster für individuelle Ikonen. Die ausgewählte individuelle Ikonen wird übernommen und im Fenster für Standard-Ikonen (Abb. 5-60) dargestellt.

Zu bemerken ist, daß die ‘**scale**’-Funktion in diesem Menü nicht für individuelle Ikonen, sondern nur für die Standard-Ikonen von PACE verwendet werden kann. Individuelle Ikonen können aber bevor sie für die Vorbesetzung verwendet werden, in dem sich bei Anwahl von ‘individual icon’ öffnenden Fenster für individuelle Ikonen mit der dort verfügbaren ‘scale’-Funktion skaliert und danach für die Vorbesetzung verwendet werden.

5.9.1.2 Individuelle Ikonen

Über das Ikonen-Fenster für individuelle Ikonen und sein Auswahlmenü werden die Ikonen eines Modelles oder eines Unternetzes

verwaltet und bearbeitet. Das Ikonen-Fenster wird über die Funktionen 'individual icons' oder 'individual module icons' geöffnet.

individual icons
Die Funktion ermöglicht dem Anwender die Definition eigener Ikonen für das in Bearbeitung befindliche Modell. Diese werden in einer Datei des Unterverzeichnis 'icons' gespeichert. Der Name der Icon-Datei wird aus dem Modellnamen und der Kennung '.icn' gebildet. Wird ein Modell mit individuellen Ikonen auf ein anderes System gebracht, muß darauf geachtet werden, daß sowohl die Netzdatei als auch alle Ikonendateien der im Modell eingesetzten Module übertragen werden.

individual module icons
Die Funktion ermöglicht dem Benutzer die Definition eigener Ikonen für den selektierten Modul. Die so in einem Fenster definierten Ikonen werden in einem File im Unterverzeichnis ‘modules’ gespeichert. Der Name des Files wird aus dem Modulnamen und der Kennung '.icn' gebildet. Auf diese Weise können sowohl die Unternetze als auch deren individuelle Ikonen in andere Modelle integriert werden.

Beim Einbinden solcher Unternetze muß allerdings darauf geachtet werden, daß sich die ‘sub’- und ‘icn’-Dateien im jeweils dafür vorgesehenen Unterverzeichnis des aktuellen PACE-Verzeichnisses befinden.

Das Fenster für individuelle Ikonen

Durch die voranstehend genannten Menüfunktionen wird das sog. Ikonenfenster geöffnet (Abb. 5-61). Es enthält eine Auflistung aller Ikonen des Modells oder selektierten Moduls.



Abb. 5-61: Das Ikonenfenster mit dem No-Selection-Menü

Das über die rechte Maustaste zu öffnende sog. No-Selection-Menü (kein Ikon ist selektiert) macht folgende Funktion zugänglich:

add

Fügt den Namen einer weiteren Ikone in die Liste ein. Diesem Namen ist dann mittels der Funktionen im Bearbeitungsmenü eine Ikone zuzuweisen.

Wurde ein Ikonen-Name angewählt und hält man die linke Maustaste niedergedrückt, so zeigt PACE die entsprechende Ikone in der rechten oberen Ecke des Ikonen-Fensters an. Die

gespeicherten Ikonen können für alle Netz-Elemente, für Marken und für Hintergrundbilder der Netzfenster verwendet werden.

Wird ein Ikon selektiert, so kann durch Drücken der rechten Maustaste das Bearbeitungsmenü für Ikonen angezeigt werden.



Abb. 5-62: Das Ikonenfenster mit selektiertem Modul und Bearbeitungsmenü

Folgende Funktionen sind darin verfügbar:

from file
Liest ein Grafik-File im bmp-Format ein und weist das Bild der selektierten Ikone zu.

from screen
Weist dem selektierten Ikonen-Namen einen vom Benutzer auszuwählenden rechteckigen Bildschirmausschnitt zu. Soll eine neue Ikone eingefügt werden, so ist zunächst mit der oben beschriebenen 'add'-Funktion ein Ikonen-Name zu vereinbaren und zu selektieren.

Nach Aufruf der 'from screen'-Funktion wird der Cursor als Kreuz dargestellt, mit dem bei gleichzeitigem Drücken der linken Maus-Taste um einen Bildschirmbereich ein Rechteck gezogen werden kann. Durch Loslassen der Maus-Taste wird der umrahmte Bereich der selektierten Ikone zugewiesen.

from clipboard
Kopiert den Inhalt des Clipboards in die Ikone, sofern die Arbeitsplattform diese Operation zur Verfügung stellt. Auf diese Weise kann eine Ikone von anderen Applikationen (z.B. Paint Shop Pro, Corel, usw.) importiert werden. Der Benutzer kann so mit externen grafischen Editoren neue Ikonen entwerfen oder existierende verändern.

to clipboard
Kopiert die Abbildung der Ikone ins Clipboard, sofern die Arbeitsplattform diese Operation unterstützt.

icon editor
Ruft den Ikonen-Editor für die selektierte Ikone auf, der in einem eigenen Abschnitt beschrieben wird (siehe weiter unten).

border width
Definiert die Breite des schwarzen Rahmens für die selektierte Ikone. Wird diese auf 0 (Null) gesetzt, erhält die Ikone keine Umrandung.

rename
Ermöglicht es, der selektierten Ikone einen neuen Namen zuzuweisen.

remove
Löscht die selektierte Ikone aus der Liste.

properties
Öffnet ein Fenster, in dem Eigenschaften der Ikone angezeigt werden. Es wird der Palettentyp (mapped oder fixed) und die verwendete Farbtiefe (Anzahl der Bits, die zur Darstellung der Farbe verwendet werden) ausgegeben.

fade

Diese Funktion wurde implementiert, um die Verwendung von Ikonen als Hintergrundbilder in Netzfenstern zu erleichtern. Sie kann für Bilder mit mapped-Paletten oder mit fixed-Paletten und Farbtiefe 24 eingesetzt werden.

Bei farbintensiven Ikonen ist das im Vordergrund des Netzfensters dargestellte Netz schlecht zu sehen. Durch einmalige oder mehrfache Anwendung der ‘fade’-Funktion kann das Bild so gebleicht werden, daß das im Vordergrund dargestellte Netz gut sichtbar wird.

Bei der Anwendung der ‘fade’-Funktion kann ein neuer Ikon-Name vereinbart werden, damit das Original-Bild erhalten bleibt.

scale

Mit der ‘scale’-Funktion kann ein vorhandenes Ikon verkleinert oder vergrößert werden. Anzugeben ist der auf das selektierte Ikon anzuwendende Maßstabsfaktor.

Bei der Anwendung der ‘scale’-Funktion kann ein neuer Ikon-Name vereinbart werden, damit das Original-Bild unverändert bleibt.

Die ‘scale’-Funktion wird hauptsächlich zwecks Größen-Anpassung von Ikonen für die verschiedenen Netzelemente verwendet. Die Ikonen für Hintergrundbilder der Netzfenster brauchen nicht skaliert zu werden, weil sie beim Einsetzen in die Fenster automatisch angepasst werden.

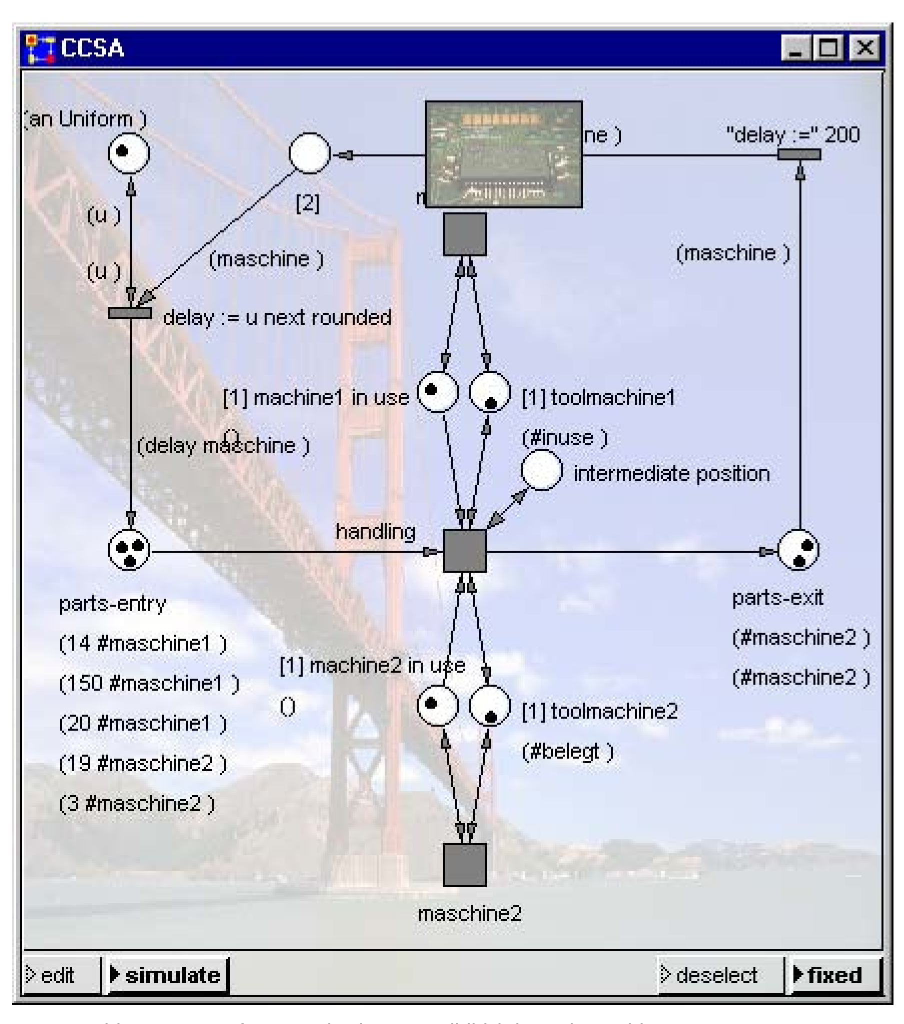


Abb. 5-63: Netzfenster mit Hintergrundbild (Photo der Golden Gate Bridge, 2 x gebleicht) und laufendes Markenikon **(**Bild eines Chip)

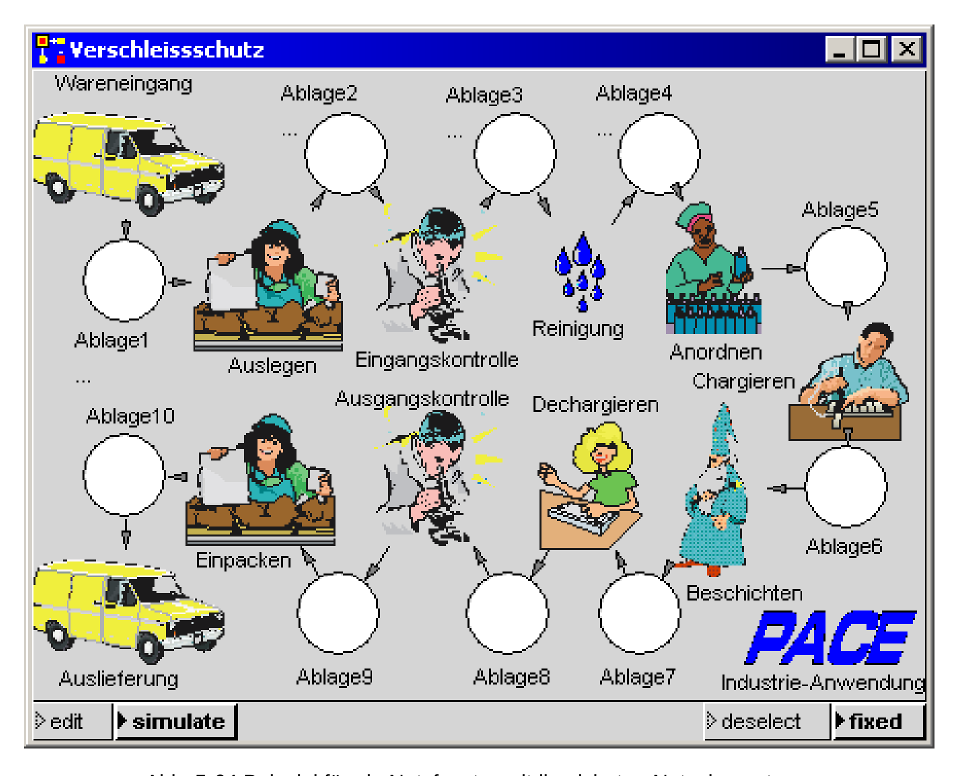


Abb. 5-64:Beispiel für ein Netzfenster mit ikonisierten Netzelementen

5.9.2 Der Icon-Editor

Der Icon-Editor ermöglicht die Neuentwicklung oder Veränderung von Grafiken, die als Hintergrundbilder oder als Ersatzbilder für die Standard-Ikonen der Netzelemente verwendet werden sollen.

Für den Aufruf des Icon-Editors ist zunächst eine Zeile in der Liste der individuellen Ikonen zu selektieren. Falls die Liste leer, muß zunächst ein Iconname mit der add-Menüfunktion (rechte Maustaste im Ikon-Fenster) eingefügt und selektiert werden.

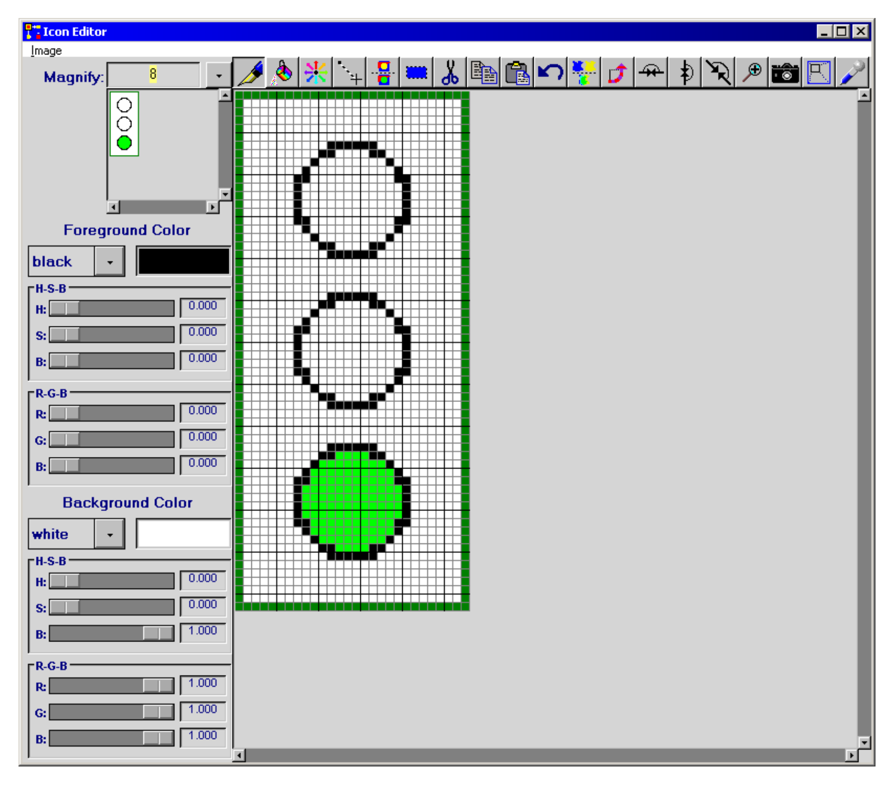


Abb. 5-65:Fenster des Icon-Editors

Dann wird über das Menü der rechten Maustaste der Ikon-Editor aufgerufen. Die angewählte Ikone wird im Icon-Editor für die weitere Bearbeitung angezeigt.

Die Oberfläche des Icon-Editors ist mehrere Felder unterteilt.

- Magnify-Feld
  Das Magnify-Feld gibt die Anzahl der Pixel an, aus denen ein (editierbarer) Bildpunkt besteht.

- Bild-Feld
  Das Bildfeld zeigt die Ikone. Die Größe entspricht der wahren Bildgröße.

- Foreground-Color-Feld
  In diesem Feld kann die Vordergrundfarbe festgelegt werden. Sie ist der linken Maustaste zugeordnet, d.h. positioniert man den Maus-Cursor auf einen Bildpunkt im Editierfenster (siehe weiter unten) und drückt die linke Maustaste, so nimmt der Bildpunkt die Vordergrundfarbe an.

  Das linke Unterfeld unter dem Schriftzug "Foreground Color" ermöglicht die Auswahl einer Standardfarbe. Bitte bachten Sie, dass bei Drücken des Auswahlknopfs im Magnify-Feld eine Zahl stehen muss.

  Rechts davon wird die aktuell festgelegte Vordergrundfarbe angezeigt.

  Mit den darunter befindlichen Schiebereglern kann eine beliebige Vordergrundfarbe in der HSB- oder in der RGB-Darstellung eingestellt werden.

- Background-Color-Feld
  wie bei der Vordergrundfarbe beschrieben. Der Hintergrundfarbe ist die rechte Maustaste zugeordnet.

- Funktionen des Ikon-Editors
  Diese werden über die Ikonen der oberer rechten Leiste aufgerufen und sind in den Unterpunkten beschrieben.

- Editierfeld
  In diesem Feld wird die Ikone vergrößert dargestellt. Die einzelnen editierbaren Bildpunkte sind durch Linien voneinander getrennt. Der Abstand der Linien ist durch die Größe der im Magnify-Feld angegebenen Zahl von Pixel zur Darstellung eines Bildpunkts bestimmt.

5.9.2.1 Menüfunktionen des Icon-Editors

Der Ikon-Editor besitzt nur einen einziges Menü: 'image' mit drei Menüfunktionen.



Abb. 5-66: Menü des Ikon-Editors

new
Erzeugt eine leere Ikone.

Bei Ausführen der Funktion öffnet sich das nachfolgend dargestellte Fenster, in dem die Anzahl der editierbaren Bildpunkte in der x- und der y-Richtung angegeben werden kann.

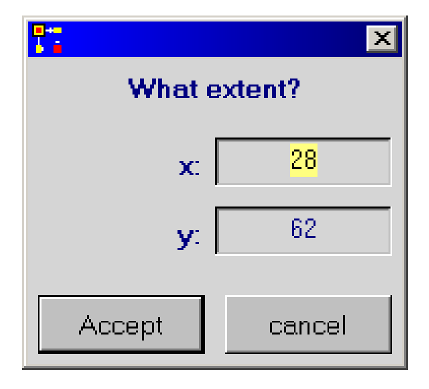


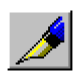
Abb. 5-67: Festlegung der Bildgröße bei einer neuen Ikone

store
Speichert die aktuell in Bearbeitung befindliche Grafik als individuelle Ikone unter dem Namen ab, der beim Aufruf des Icon-Editors selektiert war.

exit
Beendet den Ikon-Editor. Falls eine Ikone bearbeitet worden ist und die Arbeit gerettet werden soll, ist vor dem Auftruf von 'exit' die 'store'-Menüfunktion auszuführen.

5.9.2.2 Funktionen des Icon-Editor

- **Farbe selektiv einsetzen**



Schalter für selektives Einsetzen der Farben in einzelne Bildpunkte. Durch Drücken der linken / rechten Maustaste wird die Vordergrundfarbe / Hintergrundfarbe eingesetzt.

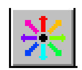
Bewegt man die Maus bei gedrückter Taste, so werden alle Bildpunkte, die dabei überstrichen werden, eingefärbt.

- **Einzelnen Bereich füllen**



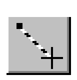
Mit der Maus wird ein Bildpunkt im Editierfenster ausgewählt. Die der gedrückten Taste zugeordnete Farbe wird in diesen Punkt und in alle Punkte seiner Umgebung, welche dieselbe Farbe haben und über eine nicht durch eine andere Farbe unterbrochene polygone Verbindungslinie in x- und/oder y-Richtung erreichbar sind, gefüllt.

- **Zusammenhängende Bereiche füllen**



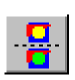
Mit der Maus wird ein Bildpunkt im Editierfenster ausgewählt. Die der gedrückten Taste zugeordnete Farbe wird in diesen Punkt und in alle Punkte seiner Umgebung, welche dieselbe Farbe haben und über eine nicht durch eine andere Farbe unterbrochene polygone Verbindungslinie in beliebiger Richtung erreichbar sind, gefüllt.

- **Gerade Linie zeichnen**



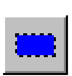
Der Cursor wird auf einen BIldpunkt, mit dem die Linie beginnen soll, positioniert. Dann die Maustaste mit der gewünschten Farbe drücken und den Cursor bei festgehaltener Maustaste auf den Endpunkt der Linie setzen. Der Editor färbt Bildpunkte so ein, dass die beste Approximation einer geraden Linie zwischen Anfangs- und Endpunkt gezeichnet wird.

- **Farbe ersetzen**



Mit der Maus wird ein Bildpunkt im Editierfenster ausgewählt. Die der gedrückten Taste zugeordnete Farbe wird in diesen Punkt und in alle Bildpunkte mit derselben Farbe gefüllt.

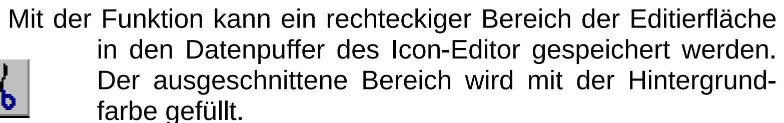
- **Rechteck zeichnen**



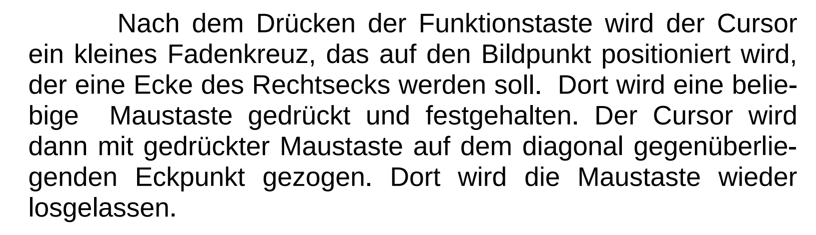
Mit der Funktion kann ein Rechteck mit der jeweils über die gewählte Maustaste festgelegten Farbe gezeichnet werden.

Man positioniert die Maustaste auf den Bildpunkt, der eine Ecke des Rechtsecks werden soll und drückt die gewählte Maustaste. Der Cursor wird daraufhin ein kleines Fadenkreuz. Der Cursor wird dann mit gedrückter Maustaste auf dem diagonal gegenüberliegenden Eckpunkt gezogen. Dort wird die Maustaste wieder losgelassen.

- **Ausschneiden**

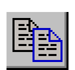


Mit der Funktion kann ein rechteckiger Bereich der Editierfläche in den Datenpuffer des Icon-Editor gespeichert werden. Der ausgeschnittene Bereich wird mit der Hintergrundfarbe gefüllt.



Nach dem Drücken der Funktionstaste wird der Cursor ein kleines Fadenkreuz, das auf den Bildpunkt positioniert wird, der eine Ecke des Rechtsecks werden soll. Dort wird eine beliebige Maustaste gedrückt und festgehalten. Der Cursor wird dann mit gedrückter Maustaste auf dem diagonal gegenüberliegenden Eckpunkt gezogen. Dort wird die Maustaste wieder losgelassen.

- **Kopieren**



Mit der Funktion kann ein rechteckiger Bereich der Editierfläche in den Datenpuffer des Icon-Editor gespeichert werden.

Nach dem Drücken der Funktionstaste wird der Cursor ein kleines Fadenkreuz, das auf den Bildpunkt positioniert wird, der eine Ecke des Rechtsecks werden soll. Dort wird eine beliebige Maustaste gedrückt und festgehalten. Der Cursor wird dann mit gedrückter Maustaste auf dem diagonal gegenüberliegenden Eckpunkt gezogen. Dort wird die Maustaste wieder losgelassen.

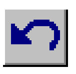
- **Einfügen**



Mit der Funktion kann ein rechteckiger Bereich, der im Datenpuffer des Icon-Editor gespeichert worden ist, über ein beliebiges gleichgroßes Rechteck des Bildbereichs gelegt werden..

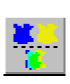
Nach dem Klicken mit der linken Maustaste auf den Funktionsknopf wird der Rahmen des einzusetzenden Rechtecks sichtbar. Er kann durch Verschieben der Maus an die Stelle des Editierbereichs gebracht werden, der überdeckt werden soll. Dort wird die Maustaste erneut gedrückt.

- **Zurücknehmen**



Die zuletzt durchgeführte Editierfunktion wird durch Drücken des Funktionsknopfs rückgängig gemacht.

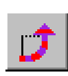
- **Mischen von Farben**



Nach dem Drücken des Funktionsknopfs öffnet sich ein Fenster zur Auswahl der Farbe, mit der die im Bild enthaltenen Farben gemischt werden sollen. Nach Auswahl der Farbe öffnet sich ein weiteres Fenster, in dem anzugeben ist, mit welcher Intensität die ausgewählte Farbe in die Mischung eingehen soll.

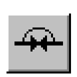
Die Farben aller BIldpunkte werden mit der ausgewählten Farbe gemischt und neu gezeichnet.

- **Drehen**



Bei Drücken des Funktionsknopfs wird das BIld um 45 Grad im Gegenuhrzeigersinn gedreht.

- **In X-Richtung spiegeln**



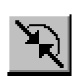
Das Bild wird an seiner vertikalen Mittellinie gespiegelt.

- **In Y-Richtung spiegeln**



Das Bild wird an seiner horizontalen Mittellinie gespiegelt.

- **In X- und Y-Richtung spiegeln**



Das Bild wird an seiner vertikalen und an seiner horizonalen Mittellinie gespiegelt. Das entspricht einer Spiegelung an der Diagonalen, die von linken unteren Eck des Bildes zum rechten oberen Eck führt oder einer zweimaligen Drehung mit der Funktion "Drehen".

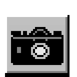
- **Bildgröße ändern**



Nach dem Drücken der Funktionstaste öffnet sich ein Fenster, in dem der Vergrößerungsfaktor in der X- und in Y-Richtung angegeben werden kann. Defaultmäßig wird für beide Richtungen der Faktor 1 angegeben.

Falls der Faktor < 1 ist, wird das Bild verkleinert. Ei n Faktor > 1 vergrößert das Bild.

- **Bildschirm fotographieren**

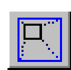


Durch diese Funktion kann ein beliebiger Bereich des Bildschirms in den Editor übertragen werden.

Nach Drücken der Funktionstaste verändert sich der Cursor in ein Fadenkreuz das durch Schieben der Maus auf eine Ecke des zu übernehmenden Bereichs gebracht wird. Dort wird

die linke Maustaste gedrückt und gehalten. Der Cursor wird dann auf die gegenüberliegende Ecke des zu übernehmenden Bereichs geschoben. Dort wird die Maustaste wieder losgelassen.

- **Ausschnitt weiterbearbeiten**



Nach Drücken der Funktionstaste verändert sich der Cursor in ein Fadenkreuz, das durch Schieben der Maus auf einen Eckpunkt des auszuschneidenden Rechtsecks positioniert wird. Dort wird die Maustaste gedrückt, gehalten und auf den gegenüberliegenden Eckpunkt des Rechtsecks positioniert, wo die Taste losgelassen wird.

Das Fenster mit dem Ausgangsbild wird gelöscht und der ausgeschnittene Bereich wird als neues Bild im Editierbereich dargestellt.

- **Pipette**



Nach dem Drücken der Funktionstaste wird der Cursor auf einen Bildpunkt im Editierbereich positioniert und eine Maustaste gedrückt. Wird die linke Maustaste gedrückt, so wird die Farbe des Bildpunkts unter dem Cursor die neue Vordergrundfarbe. Mit der rechten Maustaste wird die Hintergrundfarbe geändert.

5.9.3 Laden einer Ikon-Datei .icn

Mit der Funktion ‘load icon file’ kann eine beliebige Ikon-Datei in das in der Netzliste selektierte Netz eingelesen werden. Damit können Ikonen, die in anderen Modellen oder Netzen verwendet wurden, in das selektierte Netz eingelesen werden.

5.9.4 Szenen

Bekanntlich ist der verfügbare Platz auf dem Bildschirm bei Verwendung der zahlreichen graphischen Elemente von PACE häufig zu klein. Durch Verwendung von Szenen kann das Problem weitgehend entschärft werden. Einer Szene kann ein Satz von Fenstern zugeordnet werden, die jeweils mit einem Mausklick angezeigt oder verborgen werden können.

5.9.4.1 Vereinbarung von Szenen

Durch Ausführen der Funktion 'define scenery window' wird das in Abb. 5-68 gezeigte Definitionsfenster geöffnet, das aus zwei Teilfenstern besteht. Im linken Fenster werden die Szenen, im rechten die der aktiven Szene zugeordneten Fenster alphabetisch geordnet aufgelistet.

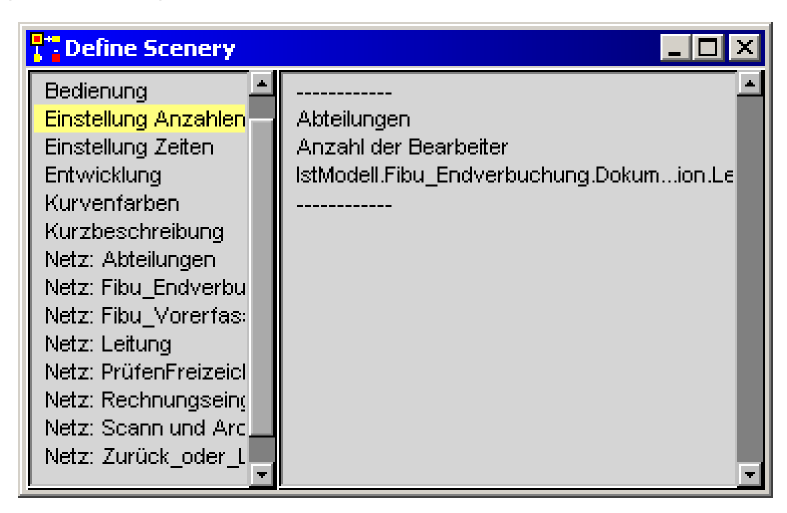


Abb. 5-68: Scenery Define Window

In beiden Teilfenstern stehen die folgenden drei Funktionen zur Verfügung:

add
Die add-Funktion wird durch Drücken der rechten Maustaste angezeigt, wenn das Fenster entweder leer ist oder wenn keine Zeile selektiert wurde.

Im linken Teilfenster wird in ein sich öffnendes Eingabefenster der Name der neu einzurichtenden Szene eingegeben. Die Eingabe ist mit der Return-Taste oder durch Ausführen der Accept-Funktion zu beenden.

Wird eine Szene durch Anklicken mit der linken Maustaste ausgewählt, so steht auch im rechten Teilfenster die add-Funktion zur Verfügung. Hier öffnet die Funktion ein Eingabefenster, in das der Name eines Fensters einzugeben ist, das der Szene zugeordnet werden soll.

rename
Mit dieser Funktion kann ein selektierter Name geändert werden.

remove
Das selektierte Element (links: Szene, rechts: Fenster) wird aus der Liste entfernt.

5.9.4.2 Selektion von Szenen



Abb. 5-69:Select Scenery Fenster mit Scenen

Das ‘define scenery’-Fenster wird nur während der Vereinbarung von Sceneries benötigt und braucht während der Arbeit mit einem Modell nicht geöffnet zu sein. Während er Arbeit mit einem Modell wird das ‘select scenery’-Fenster verwendet, mit dem die vereinbarten Sceneries angezeigt und wieder verborgen werden können.

Die Auswahl einer Szene (linke Maustaste) bewirkt, daß alle existierenden Fenster der Szene angezeigt werden. Wird die Szene wieder deselektiert, so werden alle ihr zugeordneten Fenster wieder verborgen (ikonisiert). Wird eine Szene angezeigt und eine andere ausgewählt, so wird die erstere durch die letztere ersetzt.

5.9.5 Modell-Handbuch

PACE-Modelle können für die Nutzung als sog. vergossene Modelle ausgeliefert werden (siehe Simulator-Menu). In den vergossenen Modellen verbleibt nur ein eingeschränkter Satz von meist intuitiv verwendbaren Bedienungsfunktionen, um die Modelle mit unterschiedlichen Parametern ausführen zu können.

Bei komplexeren Modellen ist es erforderlich, den Anwendern eine Modellbeschreibung an die Hand zu geben, welche die Aufgabe und Funktionsweise des Modells und seine Bedienung enthält und zusammen mit dem Modell bereitgestellt wird. Mit dem „Model User Manual“ kann eine solche Beschreibung erstellt und untrennbar mit dem Modell verbunden werden.

Damit auf Modellbeschreibungen auch in einem vergossenen Modell zugegriffen werden kann, wurden erweiterte Exekutiven mit Buttons für das Anzeigen der Bescheibung vorgesehen (siehe Simulator-Menü).

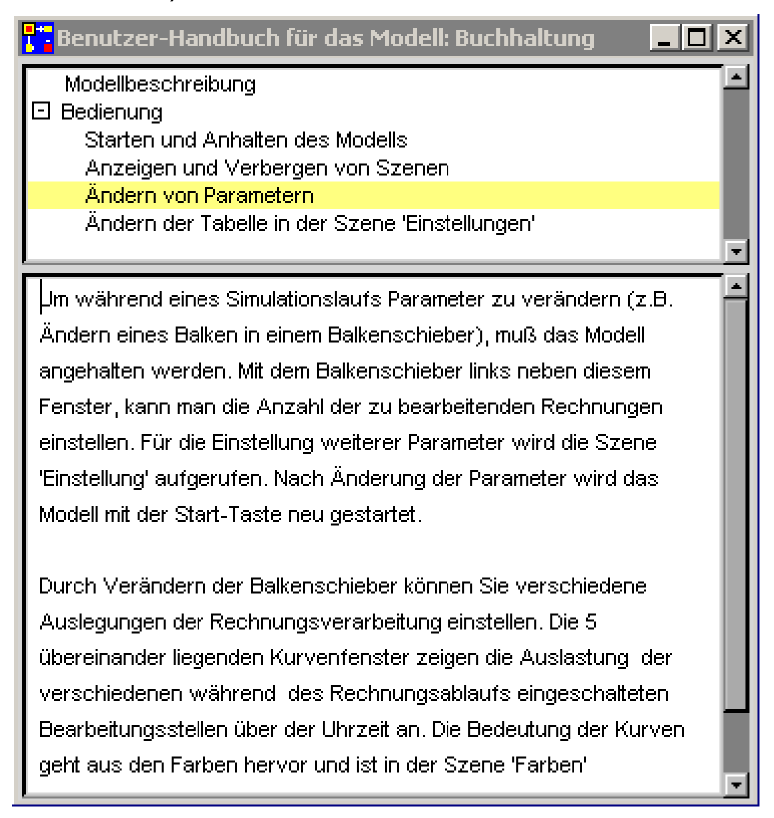


Abb. 5-70: Beispiel für ein Modell-Handbuch

5.9.5.1 Menü im Überschriften-Fenster (oberes Teilfenster)

Im Überschriften-Fenster (oberes Teilfenster) kann mit der rechten Maustaste das in Abb. 5-71 dargestellte Menü angezeigt werden.

Unter-Überschriften können verborgen und wieder angezeigt werden. Zwischen „verborgen“ und „angezeigt“ wird hin- und herge-schaltet, indem ein Doppelklick auf die nächsthöhere Überschrift ausgeführt wird. Vor dieser wird

- ein Plus-Kästchen angezeigt, wenn Überschriften verborgen sind,
- ein Minus-Kästchen angezeigt, wenn die Unter-Überschriften in der Liste der Überschriften angezeigt werden und
- nichts angezeigt, wenn keine Unter-Überschriften vorhanden sind.

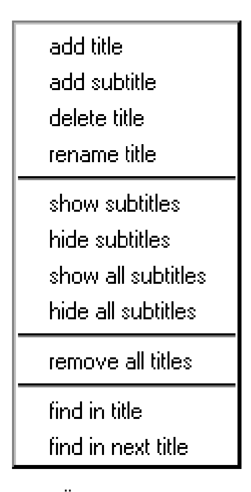


Abb. 5-71:Überschriften-Menü

Die Menü-Funktionen haben folgende Bedeutung:

add title
Fügt eine neue Überschrift auf oberste Ebene (Haupt-Überschrift) hinzu.

add subtitle
Diese Menüfunktion funktioniert nur, wenn eine Überschrift selektiert ist. Sie fügt an die Liste der Unter-Überschriften eine weitere Überschrift hinzu.

delete title
Entfernt die ausgewählte Überschrift aus der Liste der Überschriften.

rename title
Eine geänderte Überschrift kann eingegeben werden und ersetzt die bisherige Überschrift.

show subtitles
Zeigt die Unter-Überschriften der selektierten Überschrift an.

hide subtitles
Verbirgt die Unter-Überschriften der selektierten Überschrift.

show all subtitles
Zeigt die Unter-Überschriften aller Überschriften an.

hide all subtitles
Zeigt nur die Haupt-Überschriften an.

remove all titles
Löscht sämtliche Überschriften und die zugeordneten Texte.

find in title
Falls eine Überschrift selektiert ist, wird in dem zugehörigen Textfenster und dem Textfenster aller Unter-Überschriften nach dem einzugebenden Suchtext gesucht. Wird der Suchtext gefunden, so wird die Überschrift und im Textfenster der Suchtext markiert. Im Textfenster kann mit der find-Funktion des Textmenüs nach weiteren Auftreten des Suchtexts gesucht werden.

Falls keine Überschrift selektiert ist, wird der Text aller Überschriften ausgewertet. Wird der Suchtext in einem Textfenster gefunden, so wird die Überschrift und im Textfenster der Suchtext markiert.

find in next title
Ausgehend von der Überschrift, in deren Text der Suchtext gefunden wurde, wird in den Textfenstern nachfolgender Überschriften nach

dem Suchtext gesucht. Die Anzeige erfolgt wie in „find in title“ beschrieben.

5.9.5.2 Menü im Textfenster

Das Textmenü ist in der folgenden Abb. 5-72 dargestellt:

Die gezeigten Menüfunktionen wurden schon früher beschrieben.

Erwähnenswert ist, dass die accept-Funktion fehlt. Der eingegebene Text wird automatisch übernommen und bleibt beim Schließen des Fensters erhalten.

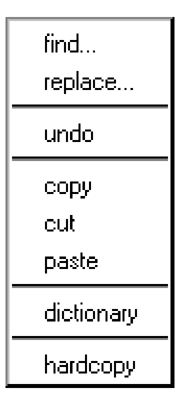


Abb. 5-72: Menü des Textfensters

Vor dem Schließen des Fensters ist die Markierung des zuletzt bearbeiteten Titels aufzuheben.

5.9.6 Zugangskontrolle

PACE-Modelle werden häufig für die Planung und Begutachtung von Geschäftsprozessen entwickelt und enthalten deshalb auch interne und geheimhaltungsbedürftige Daten und Algorithmen der jeweils modellierten Organisation. Es versteht sich von selbst, dass der Zugriff auf solche Daten nur Personen möglich sein soll, die mit der Entwicklung oder dem Einsatz eines Modells zu tun haben.

Um den Zugriff auf PACE-Modelle einzuschränken, wurde eine mehrstufige Zugangskontrolle realisiert, die bei Bedarf eingeschaltet werden kann. Es kann entweder ein einziges Passwort für den Zugang zu einem Modell vorgegeben werden oder zusätzlich eine Anzahl von Anwendern mit jeweils eigenen Passwörtern autorisiert werden.

5.9.6.1 Einrichten der Zugangskontrolle (install or change)

Das Administrator-Passwort ist nach der Installation von PACE mit einem leeren String '' vorbesetzt. Ein leeres Administrator-Passwort bedeutet, dass keine Zugriffskontrolle eingestellt ist.

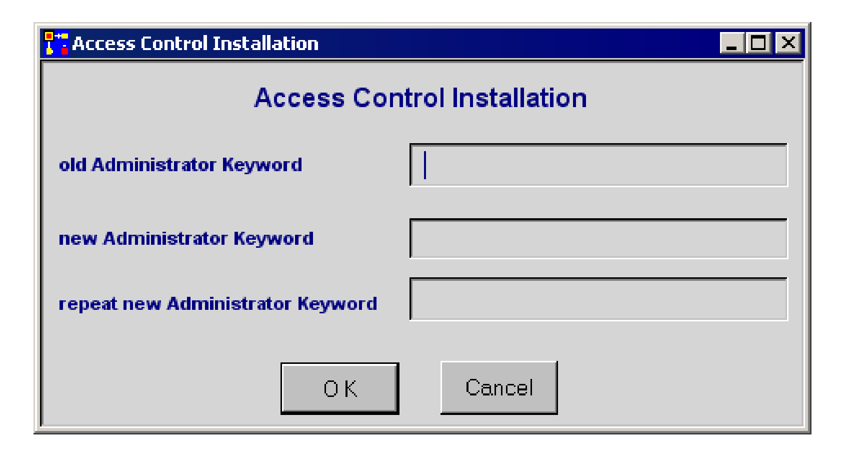


Abb. 5-73: Einrichten oder Ändern der Zugriffskontrolle

Soll der Zugriff auf das PACE-Modell eingeschränkt werden, so ist ein Administrator-Passwort in die Felder 'new Adminstrator Keyword' und 'repeat new Adminstrator Keyword' einzugeben und danach der ok-Knopf zu drücken.. Damit die vorgenommene Einstellung weiterhin gültig ist, muss das PACE-Modell nach der Eingabe abgespeichert werden.

Beim nächsten Aufruf des Image kann nur noch auf das PACE-Modell zugegriffen werden, wenn in das Login-Fenster der

Benutzername administrator und das Administrator-Passwort eingegeben wird.

Nur der Administrator kann weiteren Benutzern den Zugriff auf das Modell ermöglichen oder den Zugriff entziehen. (siehe dazu den Menüpunkt 'user control').

5.9.6.2 Benutzerverwaltung (user control)

Mit diesem Fenster kann der Administrator Anwendern den Zugriff auf das Modell ermöglichen oder entziehen. Die angezeigten Kommandos der rechten Maustaste sind selbsterklärend.

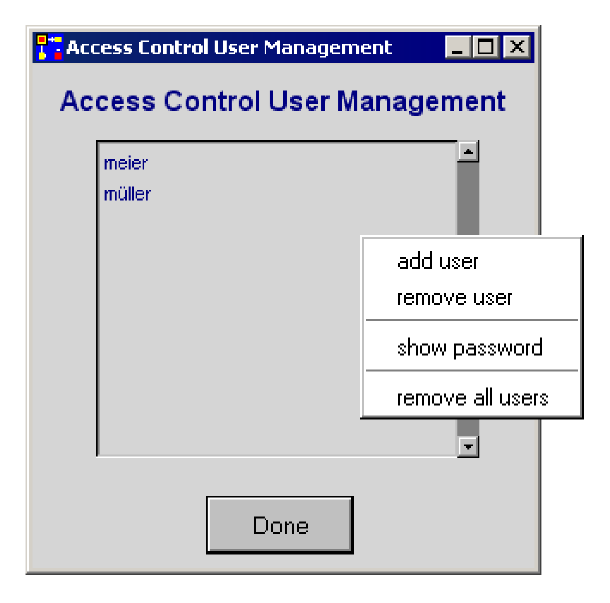


Abb. 5-74:Hinzufügen und Entfernen von Benutzern

Ist man als Administrator eingelogt, so wird nach Drücken der rechten Maustaste im Listenfeld des nebenstehend abgebildeten Fensters in Menü angezeigt, mit dem Namen und individuelle Passwörter für weitere Anwender vorgegeben werden können.

Danach muss das Modell abgespeichert werden, damit diese Einstellungen gesichert werden.

Statt des Administrators kann in das Login-Fenster, das sich beim Starten eines Modells öffnet, auch der Name und das Passwort eines eingetragenen Anwenders eingegeben werden. Dieser hat nur hinsichtlich der PACE-Features die gleichen Rechte und Möglichkeiten wie der Administrator. Er kann aber das nebenstehende Fenster nur anzeigen und nicht bearbeiten, d.h. keine Benutzer hinzufügen oder entfernen..

5.10 Window-Menü



Abb. 5-75: Window-Menü

5.10.1 Fenster neu zeichnen

refresh all

baut alle Fenster neu auf.

5.10.2 Fensterlisten

Sind in einem Modell zahlreiche Fenster geöffnet, so kann das Auffinden bestimmter Fenster, die entweder ikonisiert vorliegen oder von anderen Fenstern verdeckt sind, Probleme bereiten. Die folgenden Menüpunkte ermöglichen das Anzeigen von Fensternamen in Listen. Wird ein Element der Liste selektiert, so wird das zugeordnete Fenster angezeigt und aktiviert.

editor net windows
simulator net windows

Zeigt eine Liste der Fenster an, die geöffnet sind und sich im Editier- bzw. Simulations-Modus befinden.

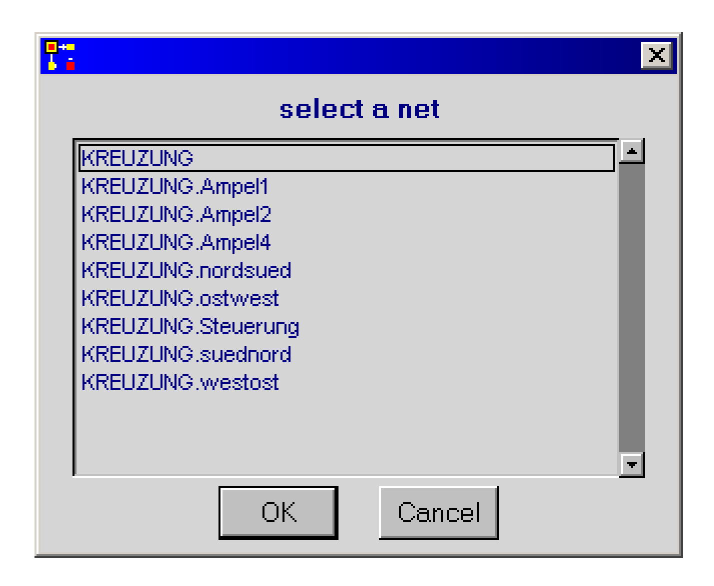


Abb. 5-76: Liste geöffneter Edit-Fenster

nets
Zeigt ein Untermenü an, von dem aus jeweils eine Liste der geöffneten Fenster angezeigt werden kann, die sich entweder im Editier-Modus oder im Simulations-Modus befinden.

button boards
Zeigt eine Liste der Fenster an, in denen jeweils eine Knopfleiste angezeigt ist.

linear gauges
Zeigt ein Untermenü an, von dem aus jeweils eine Liste der geöffneten Fenster angezeigt werden kann, die entweder einfache oder mehrfache Balkenschieberegler anzeigen.

circle gauges

Zeigt eine Liste der Fenster an, in denen jeweils ein Kreisschiebe-regler angezeigt ist.

sliders
Zeigt eine Liste der Fenster an, in denen jeweils ein Schieberegler angezeigt ist.

wheels
Zeigt eine Liste der Fenster an, in denen jeweils ein Daumenrad angezeigt ist.

curves
Zeigt ein Untermenü an, von dem aus jeweils eine Liste der geöffne-ten Fenster angezeigt werden kann, die entweder einfache oder mehrfache Kurvenfenster anzeigen

tables
Zeigt eine Liste der Fenster an, in denen jeweils eine Tabelle darge-stellt ist.

diagrams
Zeigt ein Untermenü an, von dem aus jeweils eine Liste der geöffne-ten Fenster angezeigt werden kann, die entweder ein zeitunabhängi-ges oder ein zeitabhängiges Diagramm enthalten.

5.11 Help+Info-Menü

Über das PACE-Help+Info-Menü kann das Online-Manual und können verschiedene Fenster über PACE und die einzuhaltenden Lizenzbedingungen geöffnet werden.

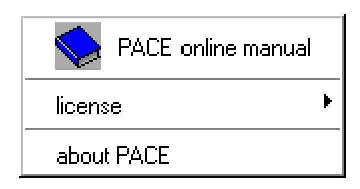


Abb. 5-77: Help+Info-Menü

5.11.1 Das PACE Online Handbuch

Der Online-Manual-Browser stellt dem Benutzer während der Arbeit mit PACE ein einfach zu handhabendes Handbuch zur Verfügung, in dem er die jeweils gewünschte Information durch Verfeinerung erreichen kann.[10] Es können beliebig viele Manual-Fenster gleichzeitig geöffnet sein. Im Online-Manual-Browser kann die jeweils gesuchte Information schnell gefunden werden. Das Online-Manual-Fenster besteht aus sechs Unterfenstern. Im oberen Teil sind die vier Titel-Fenster und im unteren Teil links das Text- und rechts das Grafik-Fenster untergebracht.

Die vier Titel-Fenster im oberen Teil enthalten eine hierarchische Auflistung der Kapitel-Titel des Manuals (von links nach rechts abfallend). Durch Aktivieren und Deaktivieren der einzelnen Titel kann man sich in den Hierarchie-Ebenen der einzelnen Kapitel bewegen.

[10] Im Gegensatz dazu wird die jeweils gewünschte Information im vorliegenden Handbuch auch über Indizierung zugänglich.

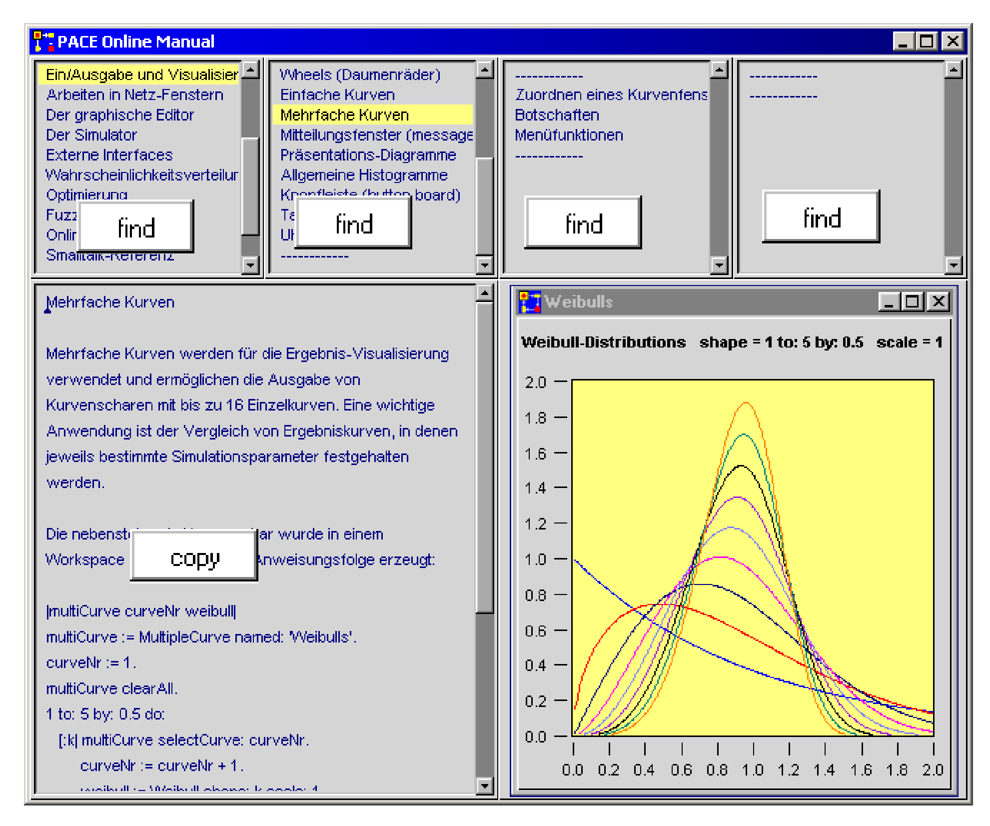


Abb. 5-78:Online-Manual mit möglichen Auswahlmenüs

Das Fenster unten links zeigt den Text des jeweils angewählten Kapitels. Im rechten Fenster können sowohl Abbildungen als auch Modelle dargestellt werden. Die Grafiken sind unveränderbar und illustrieren die textuellen Erklärungen.

Die möglichen Menüs sind in den Fenstern von Abb. 5-78 angezeigt.

find

Sucht einen Eintrag. Es wird eine Text-Eingabebox geöffnet, in welcher der zu suchende Text angegeben werden kann. Durch

Drücken der Return-Taste nach Eingabe des gesuchten Texts wird die Suche in allen Titeln gestartet.

Durch gleichzeitiges Drücken der Return- und der linken Shift-Taste wird die angegebene Zeichenfolge in allen Titeln und im Text gesucht. Wird die gesuchte Zeichenfolge im Text gefunden, so wird der Text im linken unteren Fenster (dem Text-Fenster) angezeigt.

Bei der Suche wird auf Groß/Kleinschreibung nicht geachtet. Es werden auch Teilstrings gefunden. Sobald der Manual-Browser den entsprechenden Eintrag im Manual gefunden hat, erscheint eine Dialog-Box und der Benutzer kann entscheiden, ob er die Suche fortsetzen will oder nicht.

copy
Kopiert den selektierten Text in den Paste-Buffer.

5.11.2 *PACE*-Lizenzen

PACE darf nur verwendet werden, wenn der Benutzer die Copyright- und Gewährleistungs-Bedingungen von IBE akzeptiert. Die Copyright-Bedingungen werden durch Ausführen der Menüfunktion ‘**license agreement’** angezeigt. Sie können auch zusammen mit den Gewährleistungs-Bedingungen durch Ausführen des Programms ‘Lizenzvertrag.pdf’, das sich im Hauptverzeichnis der PACE-CD befindet, angezeigt werden.

Nach der Installation liegt eine Demo-Version von PACE vor (max. 15 Netzelemente). Sie reicht aus, um die meisten Features von PACE auszuprobieren und insbesondere für die Durchführung des PACE-Starters. Die Dokumentation zum PACE-Starter kann nach der Installation von PACE im Windows-Startmenü durch Bedienen des entsprechenden Ikons angezeigt werden.

Will man eine Version von PACE mit mehr Netzknoten verwenden, so muß man bei IBE GmbH oder einem Distributor von IBE GmbH eine PACE-Lizenz bestellen, d.h. von IBE GmbH einen Lizenz-schlüssel anfordern. Für die Generierung des Lizenzschlüssels sind

an die IBE GmbH der Benutzername und die Organisation, für welche die Lizenz angefordert wird, zu übermitteln.

Nach Erhalt des Lizenzschlüssels von IBE GmbH wird die schon installierte Demo-Version wie folgt auf die bestellte Knotenzahl gebracht:(Abb. 5-79).

Durch Ausführen des Menüpunkt: 'help+info'-Menü, ‘license’, '**insert licence**' öffnet sich ein Eingabefenster mit drei Feldern, in die der Reihe nach die Lizenzangaben eingetragen werden. Zwischen der Feldern kann durch Drücken der tab-Taste gewechselt werden. Nach der Eingabe der Lizenzinformation wird die Lizenzeingabe durch Drücken der Return-Taste abgeschlossen.

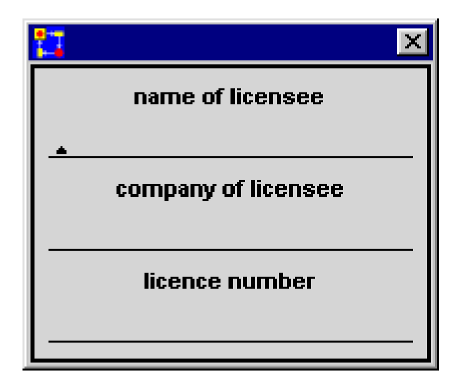


Abb. 5-79:Eingabefenster für den Lizenzschlüssel

Damit das Abspeichern des PACE-Image mit der eingetragenen Lizenz nicht vergessen wird, öffnet sich nun ein Fenster, in dem der Anwender aufgefordert wird, das Image abzuspeichern. Es wird empfohlen, das Image sofort nach Eintragen der Lizenz abzuspeichern, da nach Verlassen des Images ohne Speicherung die Lizenzinformation verloren geht und später erneut eingetragen werden muss.

Mit der Funktion 'show licence' kann bei einer lizensierten PACE-Installation der Name des Lizenznehmers und die Organisation, der er angehört, angezeigt werden.

5.11.3 Über PACE

Die Funktion ‘about PACE’ zeigt die Release-Angaben und IBE-Adressen an.



Abb. 5-80: about PACE

Bitte setzen Sie sich mit uns über eine der angegebenen Adressen in Verbindung, wenn Sie irgendein Problem mit Ihrer Installation von PACE haben, wenn Sie Fragen zu PACE und seiner Anwendung haben oder Information über neue PACE-Versionen wünschen. Informationen über neue PACE-Versionen und neue Modelle finden Sie auch auf unserer Web-Site.

6 EIN/AUSGABE UND VISUALISIERUNG

Die folgenden Abschnitte beschreiben die zahlreichen textuellen und graphischen Ein/Ausgabe-Möglichkeiten von PACE. Die standardmäßig in Smalltalk vorgesehene File-Ein/Ausgabe ist im beigefügten Smalltalk-Primer beschrieben. Spezielle Ausgabefenster für zeitabhängige Funktionen (zeitabhängige Histogramme und Liniendiagramme) werden in Abschnitt 8.8 behandelt.

6.1 Bar Gauge (Balken-Schieberegler)

Bar Gauges (oder Balken-Schieberegler) werden für das Ändern und das Anzeigen von einzelnen Werten benutzt. Sie werden normalerweise eingesetzt, um Variablenwerte zu verändern, können aber auch an bestimmte Objekte (häufig Marken-Attribute) geknüpft werden.

6.1.1 Erzeugen eines ‘Bar Gauge’s

Ein 'Bar Gauge'-Fenster wird durch Ausführen der Funktion ‘bar gauge’ im ‘view’-Menü der PACE-Hauptleiste erzeugt. Mit der folgenden Botschaft kann ein bestimmtes ‘Bar Gauge’-Fenster von Inskriptionen her zugeordnet werden:

aBarGaugeValue := BarGaugeValue named: 'name'

Beim Senden dieser Botschaft muß der benannte Bar Gauge’ (im Beispiel Abb. 6-1: 'Valve') bereits existieren. Der Wert eines Bar Gauges kann per Programm oder mit der Maus verändert werden.

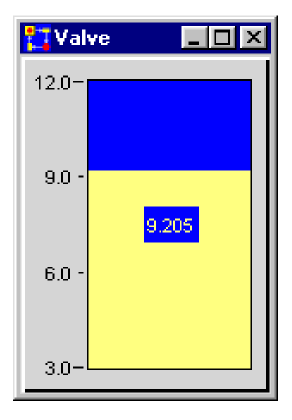


Abb. 6-1: Ein 'Bar Gauge'-Fenster

6.1.2 Botschaften an Bar Gauges

‘Bar Gauges’ können folgende Botschaften verstehen und verarbeiten:

aBarGaugeValue value.

liefert als Rückgabewert den aktuellen Wert der Instanz der Klasse 'BarGaugeValue'.

aBarGaugeValue value: aNumber.

setzt einen neuen Wert für die Instanz der Klasse 'BarGaugeValue'. Der Wert kann nur gesetzt werden, wenn der dem Balkenschieberegler zugeordnete 'block' (siehe weiter unten) der Defaultblock ist.

aBarGaugeValue barColor.

liefert die Farbe des Balken als Smalltalk-Symbol (z.B. #blue).

aBarGaugeValue barColor: aColorAsSymbol.

verändert die Farbe des Balkens in die angegebene Farbe.

Beispiel:
```
|gauge|
gauge := BarGaugeValue named: 'Aktueller Messwert'.
gauge barColor: #magenta.
```

6.1.3 Bar Gauge-Menüs

In einem Bar Gauge-Fenster stehen zwei Menüs zur Verfügung.

6.1.3.1 Menü für die Skalierung

Das Menü zur Anzeige des Darstellungs-Maßstabs wird angezeigt, wenn der Cursor links neben dem Balken positioniert und danach die rechte Maustaste gedrückt wird.

Folgende Funktionen sind verfügbar:

maximum value
Setzt den Maximalwert des Darstellungsmaßstabs.

minimum value
Setzt den Minimalwert des Darstellungsmaßstabs.

inscriptions
Setzt die Anzahl der Teilstriche.

6.1.3.2 Parameter-Menü

Das Menü für die Parametrierung des Bar Gauge erscheint durch Drücken der rechten Maustaste, wenn sich der Cursor innerhalb des für den Balken vorgesehenen Bereiches befindet.

Es ist in Abb. 6.2 abgebildet und macht folgende Menübefehle verfügbar:

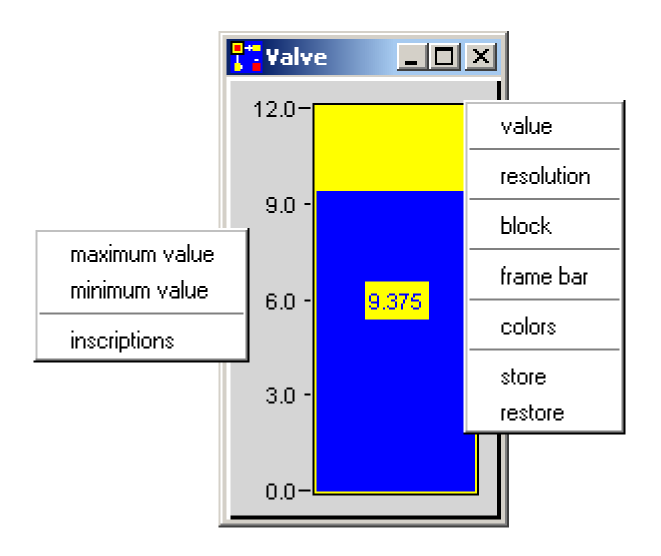


Abb. 6-2: Bar Gauge-Fenste mit dem Menü für
die Bearbeitung des Balken.

value
Setzt einen spezifischen Wert und justiert den Balken. Dieser Befehl ist vor allem dann sinnvoll, wenn nach Einstellung einer hohen Auflösung, bestimmte Werte neu gesetzt werden müssen.

resolution
Ändert die Auflösung.

block
Gibt dem Benutzer die Möglichkeit, einen Block zu definieren, der bei jeder Änderung des Balkenwertes ausgeführt wird. Der Block erhält als sein einziges Argument den Balkenwert v. In der Regel wird ein Block eingesetzt, um den aktuellen Balkenwert einer globalen Variablen zuzuweisen:

[:v | GlobalVariable := v].

frame bar

Ändert die Fläche, die der Balken innerhalb des Fensters einnimmt. Diese Funktion kann eingesetzt werden, um die Sichtbarkeit der Skalenwerte an die jeweils geforderten Genauigkeit anzupassen.

colors
Öffnet das in Abb. 6-3 dargestellte Farbauswahlfenster, über das der Balkenhintergrund und der Balken eingefärbt werden.

store
Speichert alle Parameter des Balkens in eine Textdatei. Diese Datei wird im Verzeichnis 'ioutils' gespeichert.

restore
Lädt die zuvor gespeicherten Parameter des Balkens aus einer Textdatei im Verzeichnis 'ioutils'. Nach Ausführen dieser Funktion erscheint eine Auflistung aller Dateien mit gespeicherten Balkenwerten im Verzeichnis 'ioutils'. Aus ihnen kann die gewünschte Datei ausgewählt werden.



Abb. 6-3: Farbauswahlfenster

6.2 Alternativer Bar Gauge

Der ‘Alternative Bar Gauge’ unterscheidet sich von dem in Abschnitt 6.1 beschriebenen ‘Bar Gauge’ im wesentlichen durch andere Parametrierungsmöglichkeiten.

6.2.1 Erzeugen eines ‘Bar Gauge’s

Ein 'Bar Gauge'-Fenster wird durch Ausführen der Funktion ‘alternative bar gauge’ im ‘view’-Menü der PACE-Hauptleiste erzeugt. Mit der folgenden Botschaft kann ein bestimmtes Fenster, in dem ein alternativer Bar Gauge dargestellt ist, an ein Netz angeschlossen werden:

aBarGaugeValue := AlternativeBarGauge named: 'name'

Beim Senden dieser Botschaft muß der Bar Gauge’ (im Beispiel Abb. 6-4: 'Valve') bereits existieren. Der Wert eines Bar Gauges kann per Programm oder mit der Maus verändert werden.

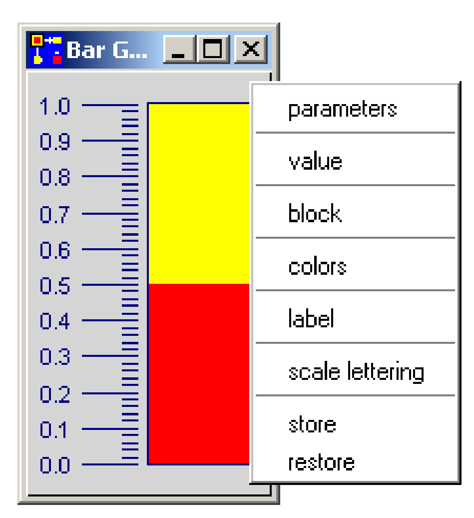


Abb. 6-4: Alternativer Bar Gauge

6.2.2 Botschaften an alternative Bar Gauges

‘Alternative Bar Gauges’ können folgende Botschaften verstehen und verarbeiten:

anAlternativeBarGauge value.

liefert als Rückgabewert den aktuell angezeigten Wert.

anAlternativeBarGauge value: aNumber.

setzt einen neuen Wert für den Bar Gauge. Der Wert kann nur gesetzt werden, wenn der dem Balkenschieberegler zugeordnete 'block' (siehe weiter unten) der Defaultblock ist.

anAlternativeBarGauge barColor.

liefert die Farbe des Balken als Smalltalk-Symbol (z.B. #blue).

anAlternativeBarGauge barColor: aColorAsSymbol.

verändert die Farbe des Balkens in die angegebene Farbe.

Beispiel:
```
|gauge|
gauge := BarGaugeValue named: 'Aktueller Messwert'.
gauge barColor: #magenta.
```

anAlternativeBarGauge label.

liefert den Label des Balkenfensters als String.

anAlternativeBarGauge label: aString.

speichert in den Label des Balkenfensters die angegebene Zeichenkette aString. Wird der leere String '' angegeben, so wird ein vorhandener Label gelöscht. Ist aktuell kein Label angezeigt, so bewirkt die Ausgabe eines leeren Labels nichts.

```
Beispiel:   gauge|
            gauge := AlternativBarGauge named: 'Aktueller Messwert'.
            gauge label: 'Meßwert um 12:02 Uhr'.
```

6.2.3 Bar Gauge-Menü

Durch Drücken der rechten Maustaste im ‘Alternative Bar Gauge’-Fenster wird das in Abb. 6-4 dargestellte Menü angezeigt.

parameter
Führt man die Funktion ‘parameters’ aus, so öffnet sich ein Definitonsfenster für die Parameter (Abb. 6-5). In ihm können der Textstil, die Skalierung, die Auflösung und die Rundung von Werten festgelegt werden. Weiter kann bestimmt werden, ob die Skala links oder rechts vom Balken angeordnet werden soll.

value
Setzt einen spezifischen Wert und justiert den Balken. Dieser Befehl ist vor allem dann sinnvoll, wenn nach Einstellung einer hohen Auflösung, bestimmte Werte neu gesetzt werden müssen.

block
Gibt dem Benutzer die Möglichkeit, einen Block zu definieren, der bei jeder Änderung des Balkenwertes ausgeführt wird. Der Block erhält als einziges Argument den Balkenwert v. Meist wird der Block eines Balkenschieberegler eingesetzt, um den aktuellen Balkenwert einer globalen Variablen zuzuweisen:

[:v | GlobalVariable := v].

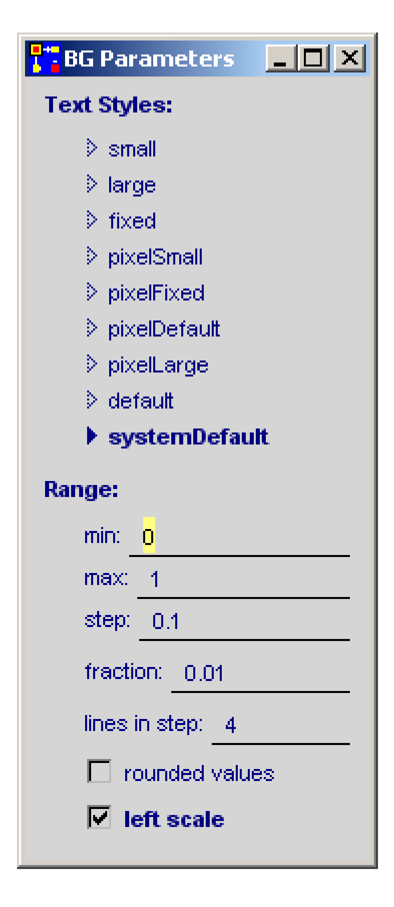


Abb. 6-5: Parameter-Fenster des ‘alternative bar gauge’

colors
Öffnet das in Abb. 6-3 dargestellte Farbauswahlfenster, mit dem der Balkenhintergrund und der Balken eingefärbt werden.

label
Außer dem Windows-Label, der zur Identifizierung des ‘Alternative Bar Gauge’-Fensters verwendet wird, kann über dem Balken ein fenster-interner Label eingesetzt werden, der für zusätzliche Erläuterungen verwendet werden kann.

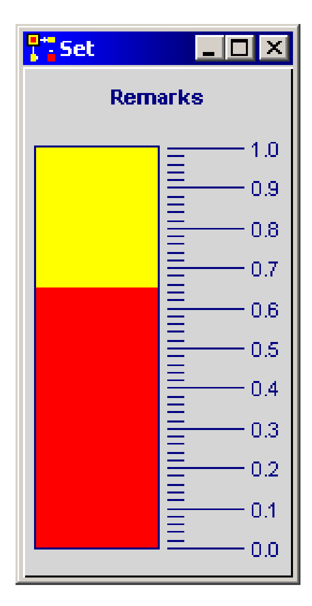


Abb. 6-6: ‘Alternative Bar Gauge’ mit Label und Skala rechts

scale lettering

Gelegentlich wird statt der numerischen Beschriftung eine alphanumerische Beschriftung gewünscht. Dabei kann jedem Step (siehe Abb. 6-8) ein Bezeichner zugeordnet werden. Sind weniger Bezeichner als Steps angegeben, so werden die Bezeichner wiederholt.

Die Bezeichner werden mit dem in Abb. 6-7 angegebenen Fenster festgelegt, in das sie der Reihe nach, getrennt durch vertikale Striche, eingegeben werden.

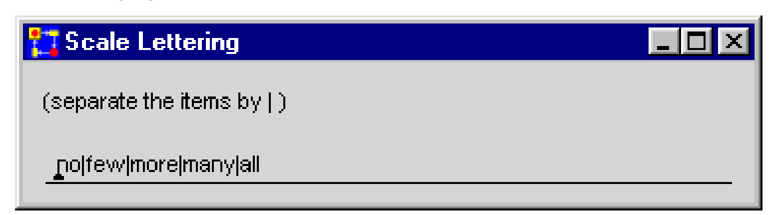


Abb. 6-7: Eingabe der Step-Namen

Die in Abb. 6-7 eingegebenen Bezeichner führen für den in Abb. 6-6 dargestellten alternativen Bar Gauge zu dem in Abb. 6-8 gezeigten Aussehen.

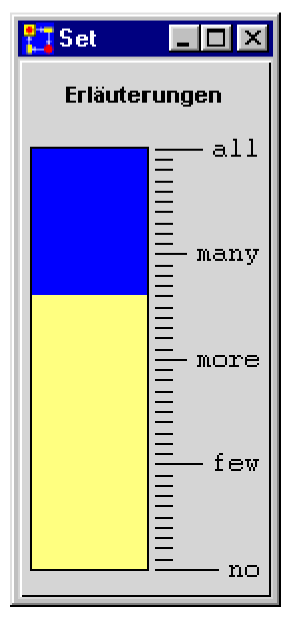


Abb. 6-8: Alternativer Bar Gauge mit alpha-numerischer Beschriftung und rechter Skala

store
Speichert alle Parameter des Balkens in eine Textdatei. Diese Datei wird im Verzeichnis 'ioutils' gespeichert.

restore
Lädt die zuvor gespeicherten Parameter des Balkens aus einer Textdatei im Verzeichnis 'ioutils'. Nach Ausführen dieser Funktion erscheint zunächst eine Auflistung aller Dateien mit gespeicherten Balkenwerten im Verzeichnis 'ioutils'. Aus ihnen kann die gewünschte Datei ausgewählt werden.

6.3 Horizontaler Bar Gauge

Der horizontale Bar Gauge entspricht bis auf die Lage des Balkens weitgehend dem alternativen (vertikalen) Bar Gauge.

Auch das Fenster für die Animationsgeschwindigkeit, das durch Anwahl des Menüpunkts 'animation speed' im Simulator-Menü der PACE-Hauptleiste geöffnet wird, kann mit den Botschaften an einen HorizontalBarGauge angesprochen werden. Zu verwenden ist die Unterklasse AnimationSpeedBarGauge der Klasse HorizontalBarGauge.

Beispiel:

```
|gauge|
gauge := AnimationSpeedBarGauge named:
                              'Animation Speed'.
```

6.3.1 Erzeugen eines horizontalen ‘Bar Gauge’s

Ein 'Bar Gauge'-Fenster wird durch Ausführen der Funktion ‘horizontal bar gauge’ im ‘view’-Menü der PACE-Hauptleiste erzeugt. Mit der folgenden Botschaft kann ein bestimmtes Fenster, in dem ein horizontaler Bar Gauge dargestellt ist, von Inskriptionen her identifiziert werden:

aBarGaugeValue := HorizontalBarGauge named: 'name'

Beim Senden dieser Botschaft, muß der horizontale Bar Gauge ‘name’ (im Beispiel Abb. 6-9: 'Horizontal Bar Gauge') bereits existieren. Der Wert eines horizontalen Bar Gauges kann per Programm oder mit der Maus verändert werden.

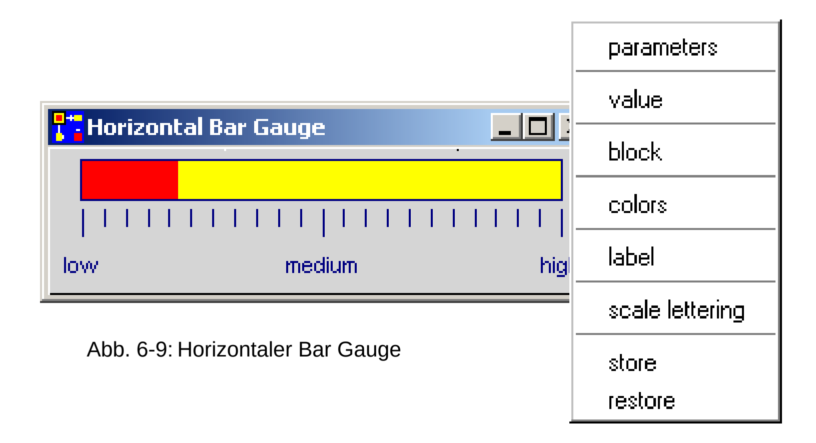


Abb. 6-9: Horizontaler Bar Gauge

6.3.2 Botschaften an horizontale Bar Gauges

‘Horizontal Bar Gauges’ verstehen und verarbeiten die gleichen Botschaften wie alternative Bar Gauges (siehe Abschnitt 6.2.2).

6.3.3 Bar Gauge-Menü

Durch Drücken der rechten Maustaste im ‘Horizontal Bar Gauge’-Fenster wird das in Abb. 6-9 dargestellte Menü angezeigt. Es ist mit dem Menü für den alternativen Bar Gauge identisch.

Auch die Menü-Funktionen sind die gleichen wie beim alternativen Bar Gauge. Im Parameter-Menü fehlt lediglich die Angabe ‘left scale’ (siehe Abb. 6-10).

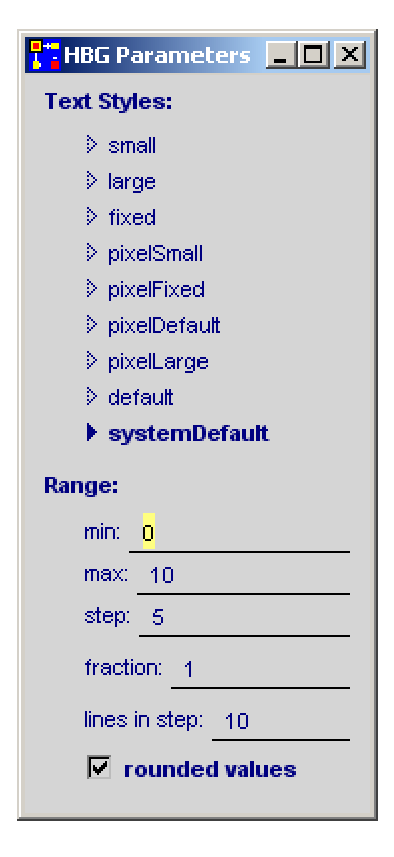


Abb. 6-10: Parameter-Fenster des ‘horizontal bar gauge’

6.4 Mehrfacher vertikaler Bar Gauge

Gelegentlich werden Bar Gauges für gleichartige Größen mit gleichem Wertebereich benötigt, die visuell miteinander ins Verhältnis gesetzt werden sollen (Beispiel: Verteilung der Arbeiter in einem Produktionsprozeß auf verschiedene Arbeitsschritte). Auch um mit dem normalerweise „immer zu kleinen” Bildschirm ökonomisch umzugehen, ist es in solchen Fällen zweckmäßig, einen ‘Multiple Vertical Bar Gauge’ zu verwenden.

6.4.1 Erzeugen eines ‘Multiple Vertical Bar Gauge’s

Ein 'Multiple Vertical Bar Gauge'-Fenster wird durch Ausführen der Funktion ‘multiple vertical bar gauge’ im ‘view’-Menü der PACE-Hauptleiste erzeugt. Beim Ausführen der Menüfunktion öffnet sich ein Fenster, in dem die Anzahl der Balken zwischen 1 und 32 angefordert werden kann (Abb. 6-11).

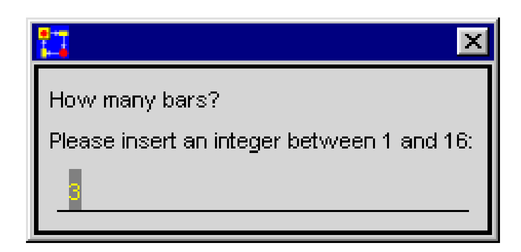


Abb. 6-11: Festlegung der Balkenzahl

Mit der folgenden Botschaft kann ein bestimmtes Fenster, in dem ein mehrfacher vertikaler Bar Gauge dargestellt ist, an ein Netz angeschlossen werden

aBarGaugeValue := MultipleVerticalBarGauge named: 'name'

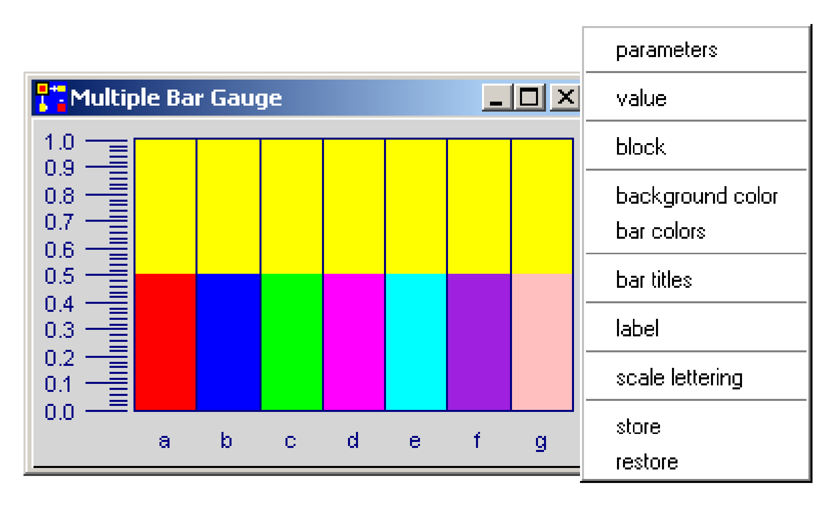


Abb. 6-12:Multiple Vertical Bar gauge mit sieben Balken

Beim Senden dieser Botschaft muß der mehrfache Bar Gauge ‘name’ bereits existieren. Der Wert eines Balkens kann per Programm oder mit der Maus verändert werden.

6.4.2 Botschaften an Bar Gauges

‘Multiple Vertical Bar Gauges’ können folgende Botschaften verstehen und verarbeiten:

aMultipleBarGauge valueAt: numberOfBar.

liefert den aktuell angezeigten Wert des Balken mit Index: numberOfBar.

.

aMultipleBarGauge valueAt: numberOfBar put: aNumber.

setzt einen neuen Wert: aNumber für den Balken mit Index: numberOfBar.

aMultipleBarGauge colorOfBar: numberOfBar.

Mit der Methode kann die aktuelle Farbe eines Balken abgefragt werden. Als Argument ist die Nummer des Balkens anzugeben. Die Farbe wird als Symbol (z. B. #blue) geliefert.

aMultipleBarGauge colorOfBar: numberOfBar
replaceBy: aNumber.

Mit der Methode kann die Farbe eines Balken verändert werden. Anzugeben ist als erstes Argument die Nummer des Balken und als zweites Argument die Farbe in Form eines Symbols.

Beispiel:
```
|gauge|
gauge := MultipleVerticalBarGauge named: 'Messwerte'.
gauge colorOfBar: 7 replaceBy: #blue.
```

aMultipleBarGauge label.

liefert den aktuellen Label des MultipleVerticalBarGauge als String.

aMultipleBarGauge label: aString.

Ändert den Label des MultipleVerticalBarGauge in den angegebenen String. Wird der leere String '' angegeben, so wird ein vorhandener Label gelöscht. Ist aktuell kein Label angezeigt, so bewirkt die Ausgabe eines leeren Labels nichts.

Beispiel:
```
|gauge|
gauge := MultipleVerticalBarGauge named:
                              'Aktuelle Messwerte'.
gauge label: 'Meßwerte um 12:02 Uhr'.
```

6.4.3 Bar Gauge-Menü

Durch Drücken der rechten Maustaste im ‘Multiple Vertical Bar Gauge’- Fenster wird das in Abb. 6-12 dargestellte Menü angezeigt.

parameter

Führt man die Funktion ‘parameters’ aus, so öffnet sich das Definitonsfenster für die Parameter (Abb. 6-13). Die Parameter sind weitgehend schon früher beschrieben worden. Zusätzlich können hier die Balken-Titel, d.h. die unter den Balken angegebene Beschriftung der Balken, an- und abgeschaltet werden.

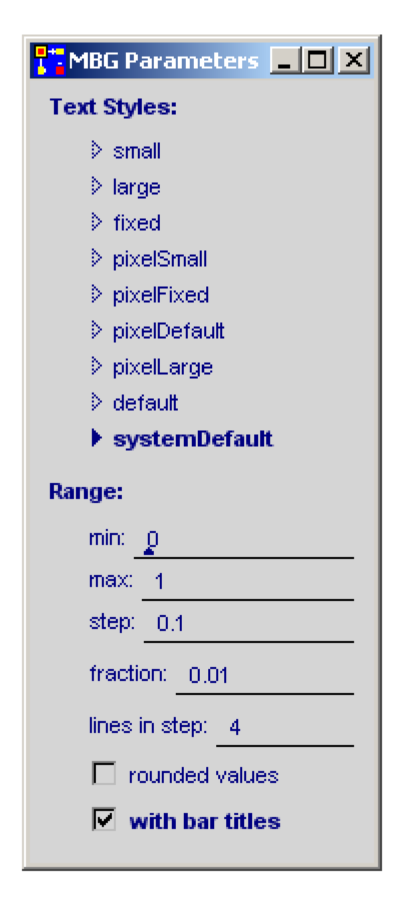


Abb. 6-13: Parameter-Fenster des Multiple Vertical Bar Gauge

value
Mit der value-Funktion kann einem Balken ein neuer Wert

zugewiesen werden. Normalerweise wird diese Funktion nur verwendet, wenn die alternative Maus-Eingabe zu ungenau ist.

Die Funktion ruft ein Eingabefenster auf, in dem der Index des zu ändernden Balkenwertes angegeben ist. Nach Eingabe des Index

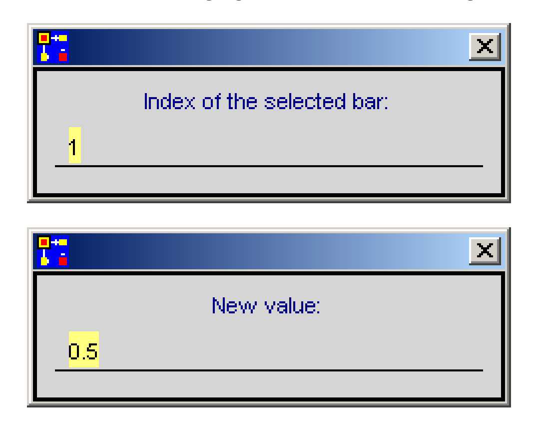


Abb. 6-14: Ändern eines Balkens

und Drücken der Return-Taste öffnet sich ein weiteres Fenster, in dem der aktuelle Wert des Balken angegeben ist. Der Balken wird geändert, indem nach Eingabe eines neuen Wertes die Returntaste gedrückt wird. Wird kein Wert eingegeben (Eingabezeile ist leer), so wird der Änderungsvorgang abgebrochen und der Balken behält seinen alten Wert.

block
Gibt dem Benutzer die Möglichkeit, einen Block zu definieren, der bei jeder Änderung des Balkenwertes ausgeführt wird. Der Block erhält als sein einziges Argument eine OrderedCollection oc mit den Balkenwerten. Häufig wird der Block dazu verwendet, um die OrderedCollection mit den aktuellen Balkenwerten einer globalen Variablen zuzuweisen:

[:oc | GlobalVariable := oc].



background color
Öffnet das in Abb. 6-15 dargestellte Auswahlfenster zur Auswahl der Grundfarbe, auf der die Balken dargestellt werden.

colors
Öffnet zunächst ein Fenster für die Auswahl des Balkens über die Balkennummer (gezählt wird von links nach rechts). Nach Absenden der Balkennummer durch Drücken der Return-Taste öffnet sich das in Abb. 6-15 dargestellte Farbauswahlfenster (die Überschrift ‘background color’ ist durch ‘bar color’ ausgetauscht), mit dem die zukünftige Farbe des Balkens bestimmt wird.

Die vorbesetzten Farben der ersten 16 Balken sind in Abschnitt 6.9.2.1 bei der Beschreibung der 'curve colors' angegeben. Ab Balken 17 werden diese Farben wiederholt, d.h. Balken 17 ist mit der gleichen Farbe vorbesetzt wie Balken 1, usw.

bar titles
Durch geeignete Balkentitel wird die Bedeutung der einzelnen Balken eines ‘Multiple Vertical Bar Gauge’s schnell ersichtlich. Der Benutzer kann deshalb analog zu der Vorgehensweise beim ‘scale lettering’ eigene Bezeichner für die Balken definieren.

Abb. 6-15: Hintergrundfarbe; die Farbe „yellow“ ist ausgewählt.

Die Funktion ‘bar titles’ öffnet ein Diagram, in dem die Bezeichner für die Balken zwischen vertikalen Strichen angegeben werden. Die Return-Taste beendet die Eingabe (siehe Abb. 6-16).



Abb. 6-16: Angabe von Bezeichnern für die Balken eines Multiple Vertical Bar Gauge

Die weiteren in Abb. 6-12 angegebenen Menüfunktionen sind in den vorangegangen Abschnitten schon beschrieben worden.

6.5 Mehrfacher Bar Gauge mit Torte

Das Verhältnis der verschiedenen Werte eines ‘Multiple Vertical Bar Gauge’s läßt sich noch bequemer übersehen, wenn den verschiedenen Balken Segmente einer Torte zugeordnet werden.

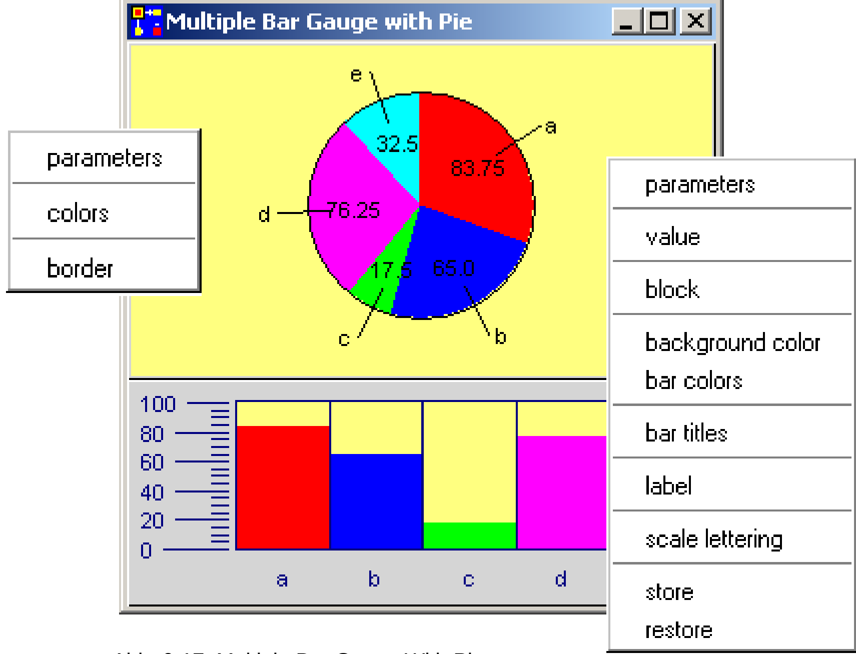


Abb. 6-17: Multiple Bar Gauge With Pie

Die in Abschnitt 6.4 beschriebenen Menüfunktionen des Multiple Vertical Bar Gauges gelten ungeändert auch hier. Bei Änderung der Balkenbezeichner (bar titles) oder der Balkenwerte werden die Angaben im Tortendiagramm und ggf. die Segmentierung der Torte automatisch angepasst.

6.5.1 Erzeugen eines ‘Multiple Vertical Bar Gauge With Pie’

Ein 'Multiple Bar Gauge With Pie'-Fenster wird durch Ausführen der Funktion ‘multiple bar gauge with pie’ im ‘view’-Menü der PACE-Hauptleiste erzeugt. Beim Ausführen der Menüfunktion öffnet sich, wie früher beschrieben, ein Fenster, indem die Anzahl der Balken zwischen 1 und 16 angefordert werden kann (Abb. 6-11).

Mit der folgenden Botschaft kann ein bestimmtes Fenster, in dem ein vertikaler Bar Gauge mit Torte dargestellt ist, an ein Modell angeschlossen werden:

aBarGaugeValue := MultipleBarGaugeWithPie named: 'name'

Beim Senden dieser Botschaft muß der Bar Gauge ‘name’ bereits existieren. Der Wert eines Balkens kann per Programm oder mit der Maus verändert werden.

6.5.2 Torten-Menü

Durch Drücken der rechten Maustaste im oberen Teilfenster eines ‘MultipleBarGaugeWithPie’-Fenster wird das in Abb. 6-17 darges-tellte Menü angezeigt.

parameters
Führt man die Funktion ‘parameters’ aus, so öffnet sich das Defini-tonsfenster für die Parameter der Torte (Abb. 6-18). Der Diagramm-Label ist schon früher beschrieben worden.

Neu ist der Toggle-Switch für das Ein- und Ausschalten der Wertan-gaben im Tortendiagramm und die optionale Angabe einer Einheit für die Werte.

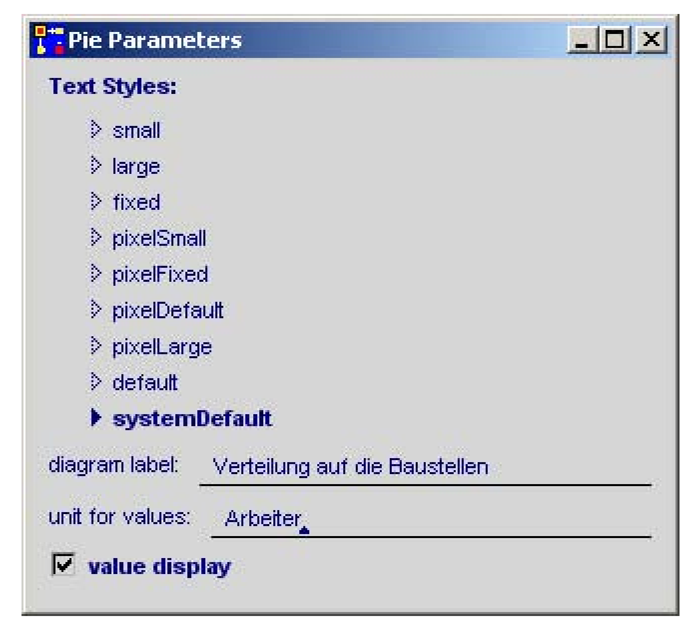


Abb. 6-18: Parameter-Fenster des Tortenteils eines ‘Multiple Bar Gauge With Pie’

colors
Es öffnet sich wieder das in Abb. 6-3 dargestellte Fenster für die Auswahl der Vorder- und Hintergrundfarben für den Tortenteil des Diagramms.

border
Um den Tortenbereich kann ein Rand gezogen werden, dessen Dicke und Farbe mit dem in Abb. 6-19 dargestellten Fenster festgelegt werden kann.



Abb. 6-19: Auswahl der Randdicke und der Randfarbe

6.6 Zeigerinstrumente

Vorgesehen sind drei Typen von Zeigerinstrumenten, die sich durch die Menüfunktionen ‘circle hand gauge’, ‘circle knob gauge’ und ‘scaleless circle knob gauge’ im ‘view’-Menü der PACE-Hauptleiste öffnen lassen. Sie entsprechen der üblichen Instrumentierung, die man häufig bei elektronischen Geräten findet. Die Knob-Instrumente können sehr klein gemacht werden und brauchen dann sehr wenig Platz auf dem Bildschirm. Alle Instrumente können sowohl für die Datenausgabe als auch für die Dateneingabe mit Maus oder über ein Eingabefenster verwendet werden.

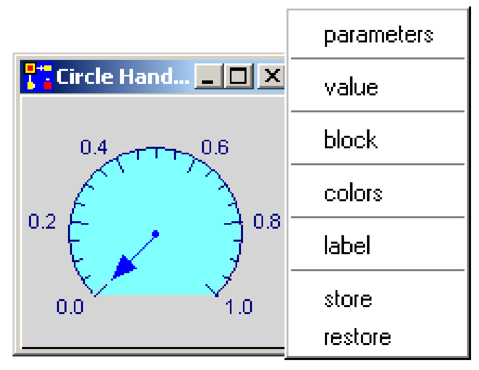
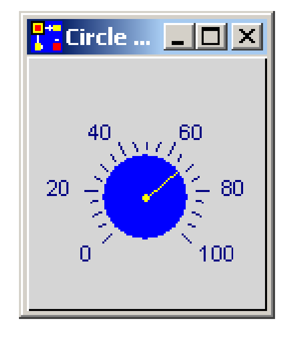


Abb. 6-20: Circle Hand Gauge



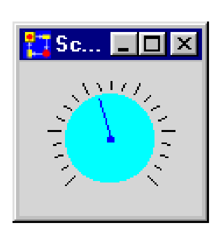


Abb. 6-21: Circle Knob Gauge und Scaleless Circle Knob Gauge

6.6.1 Zuordnen der Zeigerinstrumente

Mit einer der folgenden Botschaften kann ein bestimmtes Fenster, in dem ein Zeigerinstrument dargestellt ist, zugeordnet werden:

aCircleHandGauge := CircleHandGauge named: 'name'

oder

aCircleKnobGauge := CircleKnobGauge named: 'name'

oder

aKnobGauge := ScalelessCircleKnobGauge named: 'name'

Beim Senden einer der Botschaften muß das Zeigerinstrument mit Fensternamen ‘name’ bereits existieren.

6.6.2 Botschaften an Zeigerinstrumente

Die Botschaften sind die gleichen, die auch an ‘Alternative Bar Gauge’s gesendet werden können.

6.6.3 Circle-Menü

Durch Drücken der rechten Maustaste im Instrumentenfenster wird bei allen drei Instrumenten das in Abb. 6-20 dargestellte Circle-Menü angezeigt.

parameter
Führt man die Funktion ‘parameters’ aus, so öffnet sich ein Definitonsfenster für die Parameter (Abb. 6-22 und Abb. 6-23). In ihm können der Textstil, die Skalierung, die Auflösung und die Rundung von Werten festgelegt werden.

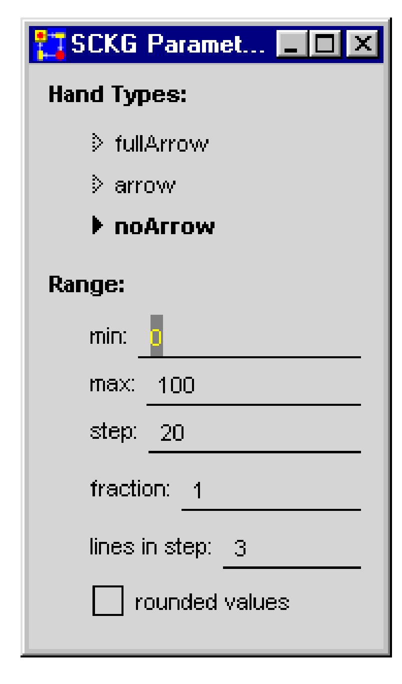


Abb. 6-22: Parameterfenster für den Scaleless Circle Knob Gauge

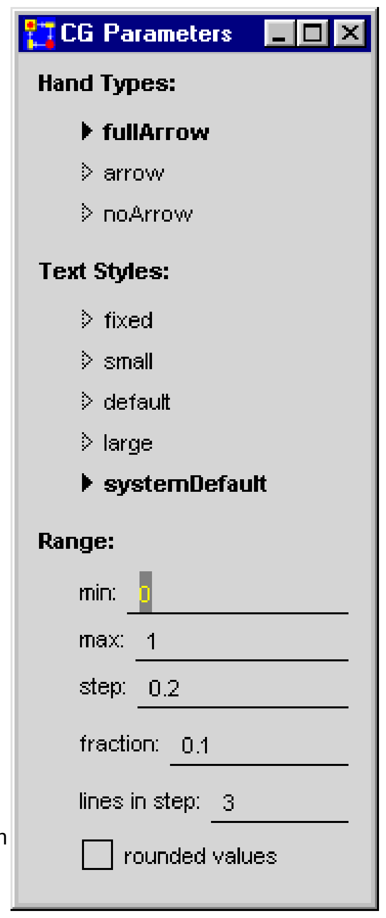


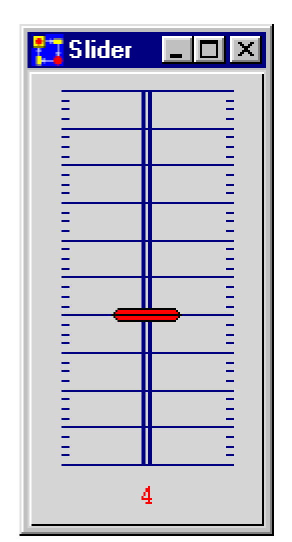
Abb. 6-23: Parameterfenster für den Circle Hand Gauge und den Circle Knob Gauge

Mit den Zeigertypen (Hand Types) können drei verschiedene Zeigerspitzen vorgegeben werden. Die übrigen Parameter sind in früheren Diagrammen schon erklärt worden.

Die weiteren Menüfunktionen sind identisch mit denen des ‘Alternative Bar Gauge’ (siehe Abschnitt 6.2) und sind dort beschrieben.

6.7 Slider (Schiebebalken)

Es gibt einen vertikalen und einen horizonalen Schiebebalken, die sich unter Verwendung der Menüfunktionen ‘vertical slider’ und ‘horizontal slider’ im ‘view’-Menü der PACE-Hauptleiste anzeigen lassen. Die Slider können sowohl für die Dateneingabe mit Maus als für die Datenausgabe verwendet werden.



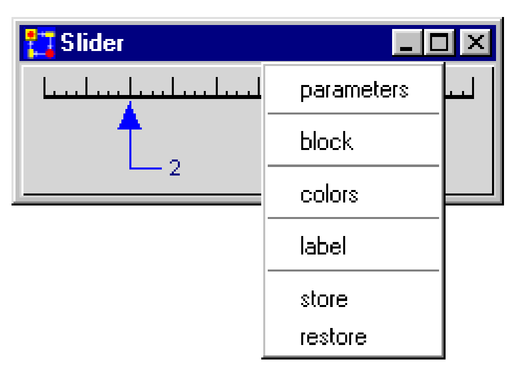


Abb. 6-24: Vertikaler und horizontaler Slider

6.7.1 Zuordnen der Slider

Mit einer der folgenden Botschaften kann ein bestimmtes Fenster, in dem ein Slider dargestellt ist, zugordnet werden:

aVerticalSlider := VerticalSlider named: 'name'

oder

aHorizontalSlider := HorizontalSlider named: 'name'

Beim Senden einer der Botschaften muß der benannte Slider bereits existieren.

6.7.2 Botschaften an Slider

Die Botschaften stimmen mit denen des ‘Alternative Bar Gauge’ überein.

6.7.3 Slider-Menü

Durch Drücken der rechten Maustaste im Sliderfenster wird bei beiden Slidern das in Abb. 6-24 dargestellte Slider-Menü angezeigt.

parameters
Führt man die Funktion ‘parameters’ aus, so öffnet sich ein Definitonsfenster für die Parameter. In ihm können der Textstil, die Skalierung, die Auflösung, die Rundung von Werten usw. festgelegt werden.

Das Parameterfenster für den vertikalen Schiebebalken ist in Abb. 6-25 dargestellt. Es enthält neben den schon früher beschriebenen auch Parameter, mit denen das Slider-Aussehen und seine Größe festgelegt werden kann.

Das Parameterfenster für den horizontalen Schiebebalken stimmt mit dem des horizontalen Bar Gauge überein (siehe Abschnitt 6.3).

Die weiteren Menüfunktionen wurden beim ‘Alternative Bar Gauge’ schon beschrieben.

VS Parameters

Slider Type:

- **polygon**
- rectangle
- wedge

width: 35

height: 7

Text Styles:

- small
- large
- fixed
- pixelSmall
- pixelFixed
- pixelDefault
- pixelLarge
- default
- **systemDefault**

Range:

min: 0

max: 10

step: 1

fraction: 1

lines in step: 3

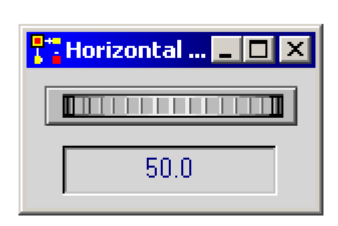
☐ rounded values

☑ **value display**

Abb. 6-25: Parameterfenster des vertikalen Slider

6.8 Wheels (Daumenräder)

Für das Einstellen von Werten gibt es ein vertikales und ein horizonales Daumenrad, die mit den Menüfunktionen ‘vertical wheel’ und ‘horizontal whell’ im ‘view’-Menü des PACE-Hauptleiste aufgerufen werden.



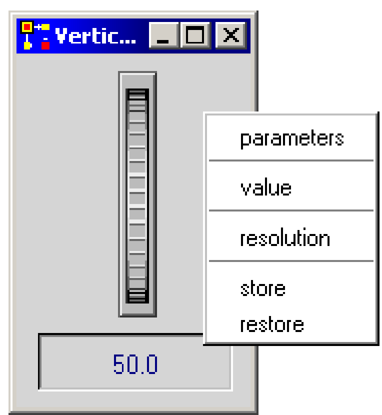


Abb. 6-26: Horizontales und vertikales Daumenrad

6.8.1 Zuordnen der Daumenräder

Mit einer der folgenden Botschaften kann ein bestimmtes Fenster, in dem ein Daumenrad dargestellt ist, von Inskriptionen her angesprochen werden:

aVerticalWheel := VerticalWheel named: 'name'

oder

aHorizontalWheel := HorizontalWheel named: 'name'

Beim Senden einer der Botschaften muß das benannte Daumenrad bereits existieren.

6.8.2 Botschaften an Daumenräder

Es gibt nur die **value-**Botschaft zum Lesen des eingestellten Wertes.

Beispiel:
```
|actualValue wheel |
wheel := HorizontalWheel named: 'ReferenceValue'.
actualValue := wheel value.
```

6.8.3 Rad-Menü

Durch Drücken der rechten Maustaste mit Cursor auf dem Daumenrad wird bei beiden Daumenrädern das in Abb. 6-26 dargestellte Daumenrad-Menü angezeigt.

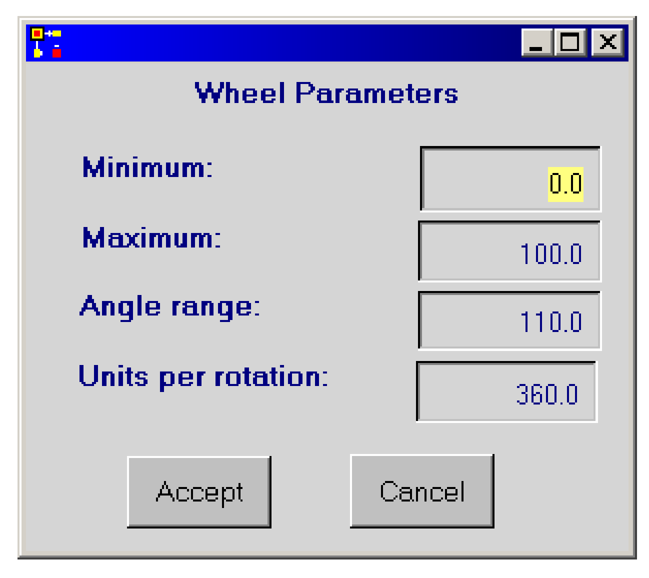


Abb. 6-27: Parameter-Definitionsfenster für Daumenräder

parameters
Führt man im Daumenrad-Menü die Funktion ‘parameters’ aus, so öffnet sich ein Definitonsfenster für die Parameter. In ihm kann der

Wertebereich, der Winkelbereich und der Wertebereich für eine volle Umdrehung des Rades angegeben werden:

Minimum:
Untere Grenze der einstellbaren Werte.

Maximum:
Obere Grenze der einstellbaren Werte

Angle range:
Wird nur ausgewertet, wenn minimum < maximum ist und gibt dann den Winkelbereich an, bei dessen Durchdrehen alle Werte angezeigt werden.

Units per rotation:
Im Fall minimum = maximum oder wenn Angle range = 0 ist, können beliebige Werte mit dem Rad eingestellt werden, d.h. es gibt keine untere und keine obere Grenze. ‘Units per rotation’ gibt den Wertebereich für eine volle Umdrehung das Rades an.

resolution
Gibt die Genauigkeit der Zahldarstellung als Dezimalzahl an.

Beispiel: 0.005 bewirkt, daß die Dezimalzahl mit einer Genauigkeit von 5 Tausendstel dargestellt wird.

store
Speichert alle Parameter des Rads in eine Textdatei. Diese Datei wird im Verzeichnis 'ioutils' gespeichert.

restore
Lädt die zuvor gespeicherten Parameter eines Rads aus einer Textdatei im Verzeichnis 'ioutils'. Nach Ausführen dieser Funktion erscheint zunächst eine Auflistung aller Dateien mit gespeicherten Radwerten im Verzeichnis 'ioutils'. Aus ihnen kann die gewünschte Datei ausgewählt werden.

6.9 Kurven

Häufig läßt sich die gegenseitige Beeinflussung von Simulationsgrößen nur ungenügend aus Momentaufnahmen, wie sie die vorangehend beschriebenen Visualisierungselemente anzeigen, erkennen. Einen vollständigeren Eindruck von dem Zusammenhang zwischen Simulationsgrößen kann man graphisch über Kurven und tabellarisch über die später beschriebenen Tabellen gewinnen.

Einfache Kurven können sowohl für die Eingabe von Daten als auch für deren Ausgabe verwendet werden. Bei der Dateneingabe kann ein bekanntes oder geschätztes Verhalten eines Simulationsparameters gezeichnet und später in die Simulation eingespielt werden.

Noch größere Bedeutung haben Kurven bei der Ausgabe von Simulationsergebnissen. Hier können mehrfache Kurven vorteilhaft eingesetzt werden. Durch Kurven wird der Zusammenhang zwischen verschiedenen Simulationsgrößen, der ggf. über mehrere Simulationsläufe hinweg aufgesammelt wird, transparent. Werte für die optimale Einstellung von Parametern der simulierten Anlage und Abhängigkeiten zwischen verschiedenen Prozeßgrößen lassen sich direkt ablesen.

6.9.1 Einfache Kurven

Ein Kurvenfenster wird durch Ausführen der Menüfunktion ‘curve’ im ‘view’-Menü der PACE-Hauptleiste erzeugt.

6.9.1.1 Zuordnen eines Kurvenfensters

Mit einer der folgenden Botschaften kann ein bestimmtes Kurvenfenster in Inskriptionen zugänglich gemacht werden:

aCurve := Curve named: 'name'

Beim Senden der Botschaft muß das Kurvenfenster ‘name’ bereits existieren.

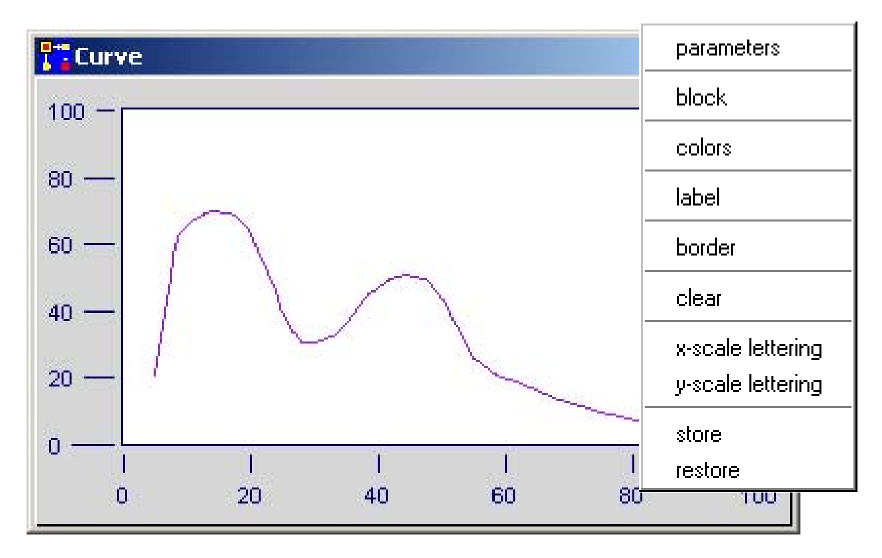


Abb. 6-28: Kurvenfenster mit Kurvenmenü

6.9.1.2 Botschaften an Kurven

Kurven können folgende Botschaften verstehen und verarbeiten:

Mit der Botschaft:

aCurve clear

wird die Kurve im Kurvenfenster gelöscht.

Die vom Benutzer oder per Programm eingegebenen Stützpunkte für eine Kurve werden in einer OrderedCollection gespeichert. Die Botschaft:

aCurve valueAt: index

liefert als Rückgabewert den ‘index’-ten Wert der Collection als Punkt x@y. Gibt es kein Element der Collection mit dem angegebenen Index, so wird der Wert nil zurückgegeben.

aCurve functionValue: x

liefert den Ordinatenwert zum Abszissenwert x. Werte zu Punkten, die zwischen den vom Benutzer bzw. per Programm definierten Stützpunkten liegen, werden durch lineare Approximation gewonnen.

Für das Einsetzen von Stützpunkten in die Collection gibt es zwei Funktionen:

aCurve increasingAt: x put: y

löscht die Kurve (falls vorhanden) ab x und fügt einen weiteren Punkt an das Ende der Kurvenbeschreibung bei x an. Die Funktion ist effizient und zu verwenden, wenn die Punkte mit zunehmenden Abszissenwerten eingegeben werden.

Kann man das nicht garantieren, so ist die zweite Eingabefunktion:

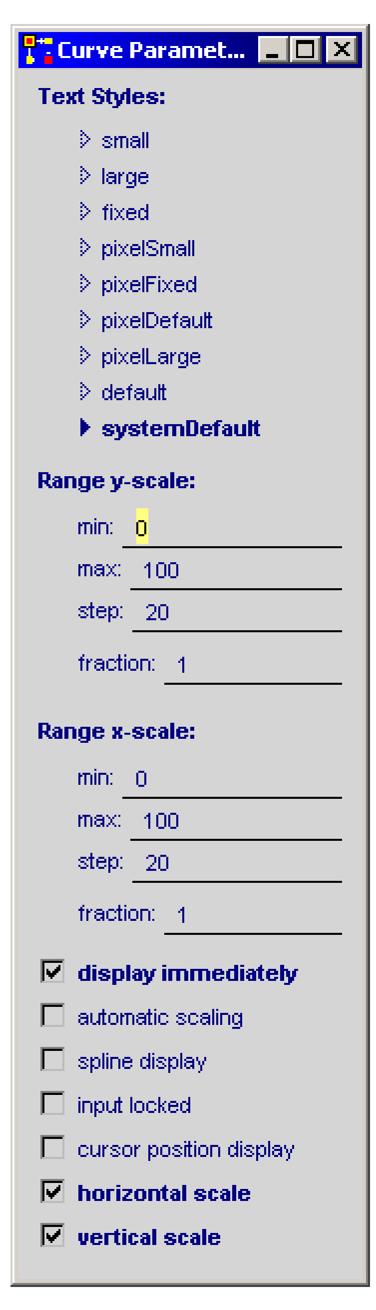
aCurve unorderedAt: x put: y

besser, die etwas mehr Rechenzeit benötigt. Sie sortiert den Punkt zwischen ggf. schon vorhandene Punkte so ein, daß die Beschreibung der Kurve nach aufsteigenden Abszissen geordnet ist.

6.9.1.3 Kurven-Menü

Durch Drücken der rechten Maustaste im Skalenbereich des Kurvenfensters wird das in Abb. 6-28 dargestellte Kurven-Menü angezeigt.

parameters
Führt man die Funktion ‘parameters’ aus, so öffnet sich ein Definitionsfenster für die Parameter (Abb. 6-29). In ihm können der Textstil, die Skalierung in x- und y-Richtung, die Auflösung in x- und y-Richtung und einige Optionen, deren Bedeutung unmittelbar aus



den Bezeichnungen im Fenster hervorgeht, ein- und ausgeschaltet werden.

block
Der angegebene Block wird bei jeder Änderung der Kurve ausgeführt. Defaultmäßig wird die Kurven-beschreibung (eine SortedCollection mit den eingegebenen Stützpunkten geordnet nach aufsteigenden Abszis-sen) in einer globalen Variablen gespeichert.

colors
label
store
restore
siehe 'Alternative Bar Gauge', Abschnitt 6.2.3.

border
siehe 'MultipleBarGaugeWithPie', Abschnitt 6.5.2.

clear
löscht die Darstellung im Kurvenfenster.

x-scale-lettering
y-scale-lettering
ersetzt die numerische Skalierung durch benutzer-definierbare Bezeich-ner. Die Skalenbezeichner werden, wie beim 'Alternative Bar Gauge' in Abschnitt 6.2.3 schon geschildert, eingegeben.

Abb. 6-29:Parameterfenster für die Kurve

6.9.1.4 Manuelle Eingabe

A) Einstellungen des Kurvenfensters

Im weiteren werden zwei Modi des Kurvenfensters unterschieden:

Menü-Mode

Im ganzen Fenster wird der Cursor als kleiner Pfeil dargestellt. Drücken der rechten Maustaste an beliebiger Stelle des Kurvenfensters zeigt das oben beschriebene Kurven-Menü an. Mit der mittleren Maustaste wird ein Menü mit den Standard-Fensterfunktionen des Betriebssystems zugänglich.

Ist eine Kurve im Kurvenbereich des Fensters dargestellt, so wird diese durch Splines approximiert.

Durch Drücken der linken Maustaste an beliebiger Stelle des Kurvenfensters kann in den Eingabe-Mode umgeschaltet werden.

Eingabe-Mode

Der Skalenbereich des Kurvenfensters ist weiterhin im Menü-Mode, der Kurvenbereich im Eingabe-Mode für eine Kurve. Im Kurvenbereich des Fensters wird der Cursor als kleines Fadenkreuz dargestellt.

Ist eine Kurve im Kurvenbereich des Fensters dargestellt, so wird diese zwischen den Stützpunkten stückweise linear approximiert.

Durch Drücken der linken Maustaste im Skalenbereich des Fensters wird in den Menü-Mode umgeschaltet.

B) Eingabe einer Kurve

Im folgenden wird vorausgesetzt, daß sich der Cursor im Kurvenbereich des Kurvenfensters befindet und der Eingabe-Mode eingestellt ist.

Die Maustasten haben dann folgende Funktionen:

Linke Maustaste

Positioniert den nächstgelegenen Stützpunkt auf den aktuellen Cursorpunkt, sofern die Kurve dabei eindeutig bleibt (jedem x-Wert darf nur ein y-Wert zugeordnet sein). Kann die Positionierung nicht durchgeführt werden, so springt das Fadenkreuz auf den nächstgelegenen Stützpunkt.

Wird die Maustaste gedrückt gehalten, so kann der Stützpunkt durch Verschieben des Cursors in dem Bereich, in dem die Kurve eindeutig bleibt, umpositioniert werden. Der Stützpunkt wird durch Loslassen der Maustaste festgeschrieben.

Mittlere Maustaste
Umrahmt den nächstgelegenen Stützpunkt mit einem kleinen roten Kreis und öffnet ein Abfragefenster mit der Frage, ob der Stützpunkt gelöscht werden soll.

Rechte Maustaste
Setzt einen neuen Stützpunkt ein. Hält man die Maustaste gedrückt, so kann der Stützpunkt durch Bewegen des Cursors in den Grenzen positioniert werden, in denen die Kurve eindeutig ist. Der Stützpunkt wird durch Loslassen der Maustaste festgeschrieben.

6.9.2 Mehrfache Kurven

Mehrfache Kurven ermöglichen die Ausgabe von Kurvenscharen mit bis zu 16 Einzelkurven. Eine wichtige Anwendung ist der Vergleich von Ergebniskurven, in denen jeweils bestimmte Simulationsparameter festgehalten werden.

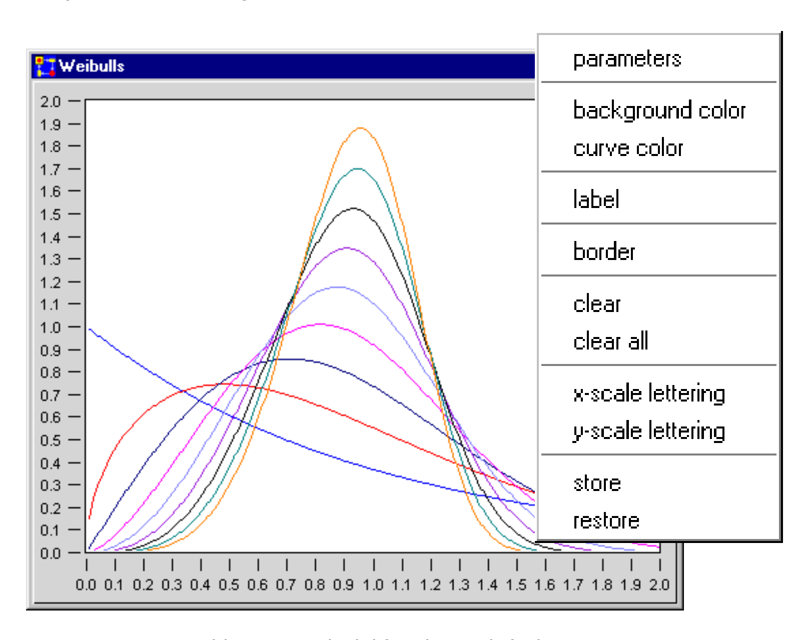


Abb. 6-30: Beispiel für eine mehrfache Kurve

Ein Beispiel für eine Kurvenschar ist in Abb. 6-30 dargestellt. Die Schar wurde mit der folgenden Anweisungsfolge in einem Workspace erzeugt:

```
|multiCurve curveNr weibull|
multiCurve := MultipleCurve named: 'Weibulls'.
curveNr := 1.
multiCurve clearAll.
1 to: 5 by: 0.5 do:
```

```
[:k| multiCurve selectCurve: curveNr.
        curveNr := curveNr + 1.
        weibull := Weibull shape: k scale: 1.
        0.01 to: 2.0 by: 0.005 do:
          [:i| | y |
            y := weibull density: i.
            (y < 3 and: [y > 0.01]) ifTrue:
               [multiCurve increasingAt: i put: y]
    ]

]
```

6.9.2.1 Zuordnen eines Kurvenfensters

Fenster für mehrfache Kurven werden mit der zugeordneten Funktion im view-Menü der PACE-Hauptleiste erzeugt. Mit der folgenden Botschaften kann ein bestimmtes, so erzeugtes Kurvenfenster für die weitere Bearbeiotung in Inskriptionen zugänglich gemacht werden:

```
aMultipleCurve := MultipleCurve named: 'name'
```

Darin ist 'name' der Name des Kurvenfensters. Beim Senden der Botschaft muß das Kurvenfenster bereits existieren.

6.9.2.2 Botschaften an mehrfache Kurven

Im folgenden werden Methoden beschrieben, über die mehrfache Kurven von Smalltalk aus angesprochen werden können.

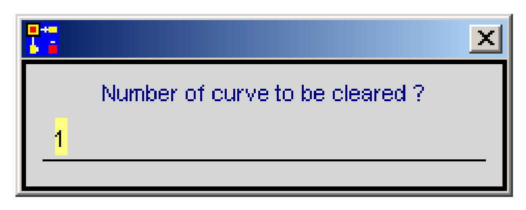


Abb. 6-31: Löschen einer einzelenen Kurve

clear
Die Funktion öffnet das in Abb. 6-31 dargestellte Fenster, in dem die Nummer der zu löschenden Kurve einzugeben ist.

Wird das Feld leer gelassen, so wird keine Kurve gelöscht.

Beispiel: multiCurve := MultipleCurve named: 'Kurven'.
multiCurve clear.

clear:
Mit dieser Funktion kann eine bestimmte Kurve unbedingt gelöscht werden:

aMultipleCurve clear: curveNumber.

Die Nummer der Kurve wird nach dem Doppelpunkt angegeben.

Beispiel: multiCurve := MultipleCurve named: 'Kurven'.
multiCurve clear: 5.

clearAll
Diese Funktion löscht alle Kurven aus dem Kurvenfenster.

Beispiel: multiCurve := MultipleCurve named: 'Kurven'.
multiCurve clearAll.

selectCurve:
Diese Anweisung gibt die Nummer der Kurve an, zu der als nächstes Punkte hinzugefügt werden sollen.

Beispiel: multiCurve := MultipleCurve named: 'Kurven'.
multiCurve selectCurve: 5.

increasingAt:put:
Mit dieser Anweisung werden Punkte in eine Kurve geschrieben. Die Kurve ist vor der ersten Ausgabe mit der Anweisung selectCurve: auszuwählen.

Beispiel: multiCurve := MultipleCurve named: 'Kurven'.
multiCurve selectCurve: 5.
multiCurve increasingAt: 5 put: 10.
multiCurve increasingAt: 15 put: 5.

drawAll
Erneute Ausgabe aller Kurven in das Kurvenfenster.

label:
Mit dieser Methode kann einem MultipleCurve-Fenster eine Überschrift hinzugefügt werden:

aMultipleCurve label: aString.

aString gibt die auszugebende Überschrift an. Ist aString der leere String '', so wird eine ggf. vorhandene Überschrift gelöscht.
Beispiel: multiCurve := MultipleCurve named: 'Kurven'.
multiCurve label: 'Kurven der Verteilungen'.

6.9.2.1 Menüfunktionen

Durch Drücken der rechten Maustaste im Fenster für mehrfache Kurven wird das inn Abb. 6-30 dargestellte Kurven-Menü angezeigt.

parameters
Führt man die Funktion 'parameters' aus, so öffnet sich ein Definitonsfenster für die Parameter (Abb. 6-32)). In ihm können der Textstil, die Skalierung in x- und y-Richtung, die Auflösung in x- und y-Richtung und einige Optionen, deren Bedeutung unmittelbar aus den Bezeichnungen im Fenster hervorgeht, ein- und ausgeschaltet werden.

Multiple Curve ...

Text Styles:

- small
- large
- fixed
- pixelSmall
- pixelFixed
- pixelDefault
- pixelLarge
- default
- **systemDefault**

Range y-scale:

min: 0

max: 100

step: 20

fraction: 1

Range x-scale:

min: 0

max: 100

step: 20

fraction: 1

- [x] **display immediately**
- [ ] automatic scaling
- [x] **horizontal scale**
- [x] **vertical scale**

Abb. 6-32: Parameter-Fenster einer mehrfachen Kurve

background color
Durch Anklicken einer Farbe im 'Color-Dialog'-Fenster (siehe Abb. 6-15) wird die Hintergrundfarbe des Fensters für mehrfache Kurven ausgewählt.

curve color
Die Funktion öffnet das in Abb. 6-33 dargestellte Fenster, in dem die Nummer der Kurve anzugeben ist, deren Farbe geändert werden soll. Durch Ausführen der 'accept'-Funktion, die über das Menü der rechten Maustaste mit Cursor im Abfragefenster erreicht wird oder durch Drücken der Return-Taste mit Cursor im Abfragefenster, öffnet sich ein 'Color-Dialog'-Fenster (siehe Abb. 6-15). Durch Anklicken einer Farbe im 'Color-Dialog'-Fenster wird die Farbe der ausgewählten Kurve geändert.



Abb. 6-33: Ändern der Farbe einer Kurve

Die 16 möglichen Kurven sind mit folgenden Farben vorbesetzt:

Kurve 1: red
Kurve 2: blue
Kurve 3: green
Kurve 4: magenta
Kurve 5: cyan
Kurve 6: purple
Kurve 7: pink
Kurve 8: orange
Kurve 9: royalBlue
Kurve 10: darkCyan
Kurve 11: orchid
Kurve 12: darkGreen

Kurve 13:	navy
Kurve 14:	salmon
Kurve 15:	brown
Kurve 16:	olive

label
siehe 'Alternative Bar Gauge'.

border
siehe 'MultipleBarGaugeWithPie'.

clear
clearAll
siehe Abschnitt 6.9.2.2

x-scale lettering
y-scale lettering
Ersetzt die numerische Skalierung durch benutzer-definierbare Bezeichner. Die Skalenbezeichner werden, wie schon beim 'Alternative Bar Gauge' geschildert, eingegeben.

store
restore
siehe 'Alternative Bar Gauge'.

6.10 Mitteilungsfenster

Über Mitteilungsfenster (‘prompt-window’s) können Informationen (Texte, Zahlen, usw.), während des Ablaufs einer Simulation angezeigt werden.

Hierzu ist zunächst ein Fenster mit der Menüfunktion ‘message window’ im ‘view’-Menü der PACE-Hauptleiste zu öffnen und durch Drücken der mittleren Maustaste und Auswahl der Fensterfunktion ‘relabel as...’ mit dem gewünschten Namen zu versehen.

Der Ausdruck

fensterName show: messageString

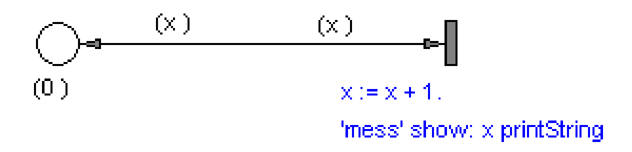


Abb. 6-34: Beispiel für die Erzeugung einer Mitteilung

innerhalb einer Transitions-Inskription zeigt die Botschaft 'message-String' in dem Mitteilungsfenster an, das den Namen 'fensterName' trägt. Sowohl der Fenstername als auch die Botschaft sind Zeichen-ketten (strings). Befindet sich kein Fenster mit dem spezifizierten Namen auf dem Bildschirm, so hat der Ausdruck keine Wirkung.

In dem Beispiel in Abb. 6-34 wird bei jedem Feuern der Transition der Wert des Marken-Attributs im Mitteilungsfenster mit dem Namen 'mess' angezeigt.

6.11 Präsentations-Diagramme

Diagramme unterscheiden sich von den früher beschriebenen graphischen Ein/Ausgabe-Elementen dadurch, daß sie für die Eingabe von Daten weniger geeignet sind, dafür aber eine besondere Ergebnisdarstellung ermöglichen, die in kaufmännischen Darstellungen häufig eingesetzt wird. Durch die in PACE vorgesehenen Balken- und Tortendiagramme (siehe auch Abschnitt 6.5) werden die verschiedenen Ausgabegrößen infolge einer automatischen Anpassung der Darstellung (Länge der Balken, Größe der Kreissegmente) in eine augenfällige Relation zueinander gesetzt.

Durch die Menüfunktionen ‘vertical bar diagram’, ‘horizontal bar diagram’ und ‘pie diagram’ im ‘view’-Menü der PACE-Hauptleiste werden, wie der Name sagt, vertikale und horizontale Balkendiagramme sowie Tortendiagramme erzeugt.

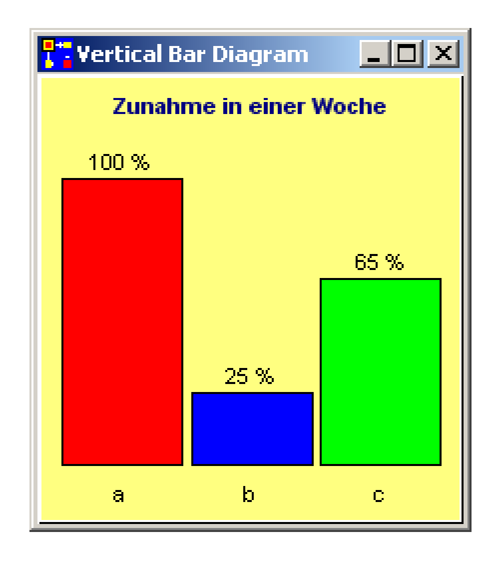


Abb. 6-35: Vertical Bar Diagram

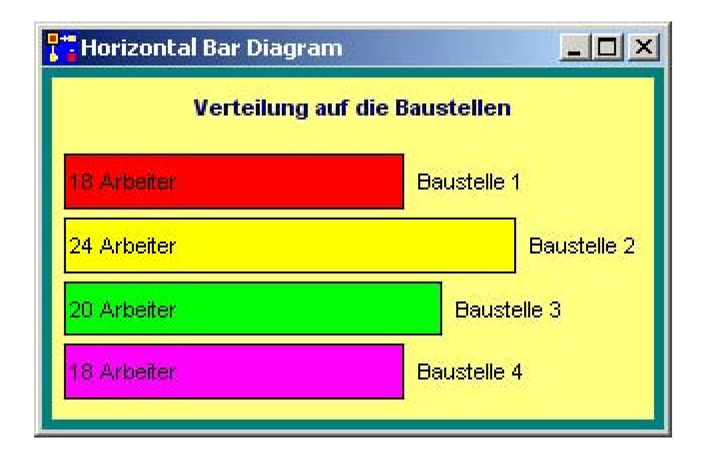


Abb. 6-36:Horizontal Bar Diagram

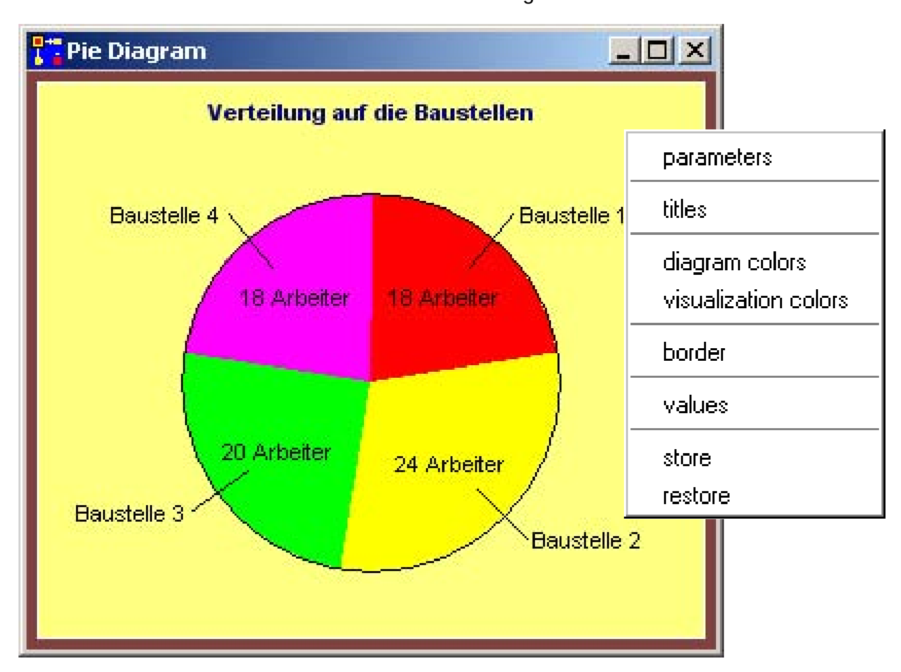


Abb. 6-37: Pie Diagram (Tortendiagramm)

6.11.1 Zuordnen der Diagramme

Mit einer der folgenden Botschaften kann ein bestimmtes Fenster, in dem ein Diagramm dargestellt ist, der weiteren Bearbeitung zugänglich gemacht werden:

aVerticalBarDiagram := VerticalBarDiagram named: 'name'

oder

aHorizontalBarDiagram := HorizontalBarDiagram named: 'name'

oder

aPieDiagram := PieDiagram named: 'name'

Darin ist ‘name’ der Name des Fensters, in dem das Diagramm gezeichnet ist. Beim Senden einer der Botschaften muß das benannte Diagramm bereits existieren.

6.11.2 Botschaften an Diagramme

Die Botschaften sind die gleichen, wie sie auch an ‘Multiple Bar Gauge’s (siehe Abschnitt 6.4) gesendet werden können.

6.11.3 Diagramm-Menü

Durch Drücken der rechten Maustaste im Diagrammfenster wird bei allen drei Diagrammen das in Abb. 6-37 dargestellte Diagramm-Menü angezeigt. Die Funktionen sind in früheren Abschnitten schon erläutert worden:

parameters
colors
border

siehe Abschnitt 6.5: Multiple Bar Gauge With Pie.

titles
values
siehe Abschnitt 6.4: Multiple Bar Gauge.

6.12 Tabellen

Bei der Erstellung von Simulationsmodellen mit PACE kann die gesamte in Smalltalk vorgesehene Standard-Ein/Ausgabe eingesetzt werden (siehe auch: Smalltalk Fibel, Kap. 5). Das ist zwar für den Entwickler eines Simulationsmodells eine notwendige, für den Anwender, der das Simulationsmodell nur benutzen will, jedoch häufig eine unzumutbare Vorgehensweise, erfordert sie doch Kenntnisse in der Programmierung mit Smalltalk, die bei der Ausführung von Modellen sonst nicht nötig sind.

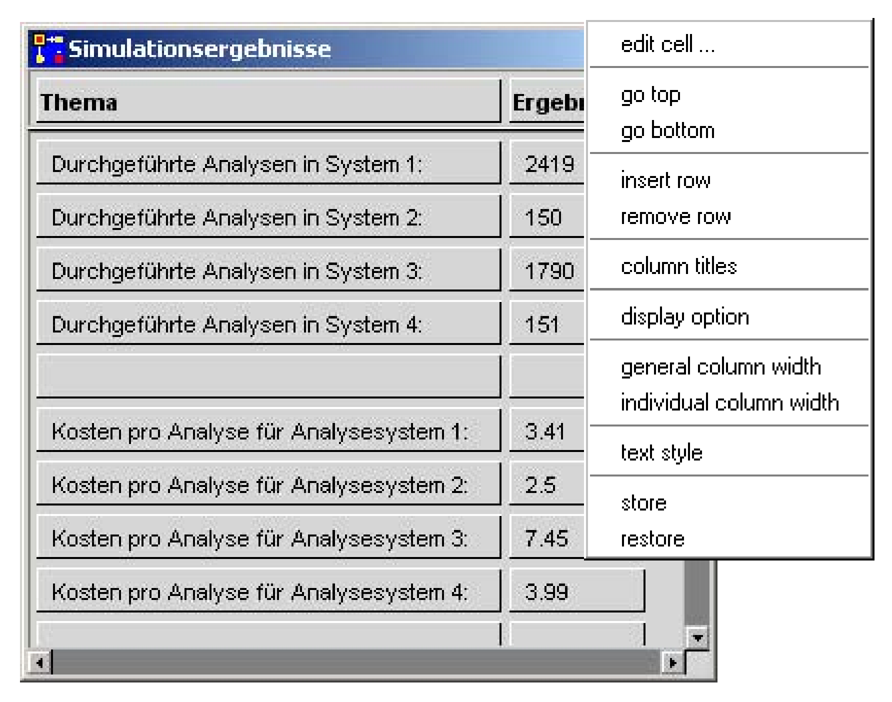


Abb. 6-38:Tabelle mit Tabellen-Menü

Deshalb wurde in PACE eine bequeme Ein/Ausgabe-Möglichkeit über Tabellen vorgesehen. PACE beschränkt sich dabei auf die reine Daten-Ein/Ausgabe; die Verarbeitung der Daten muß im Rahmen des Simulationsmodells selbst formuliert werden.

PACE-Tabellen können via DDE (siehe Kapitel: Externe Interfaces) in SpreadSheet-Programme (z.B. MS-Excel, Lotus 1-2-3) überführt, SpreadSheet-Tabellen in PACE eingelesen werden. Einerseits stehen damit die per Simulation gewonnenen Daten für die weitere Verarbeitung im Rahmen von SpreadSheets zu Verfügung. Andererseits erweitert PACE damit das Spektrum von SpreadSheet-Programmen; überall dort, wo die mit SpreadSheets durchführbare eingeschränkte Simulation (Durchrechnen von Formeln für verschiedene Parameter) nicht hinreicht, können die Daten in einfacher Weise nach PACE transferiert und dort für echte, beliebig komplexe Simulationsläufe verwendet werden.

Durch Ausführen der Menüfunktionen ‘table’ im View-Menü des Visual Net Developers öffnet sich das in Abb. 6-39 dargestellte Abfragefenster, in dem zunächst die Dimensionen der zu erzeugenden Tabelle einzugeben sind.

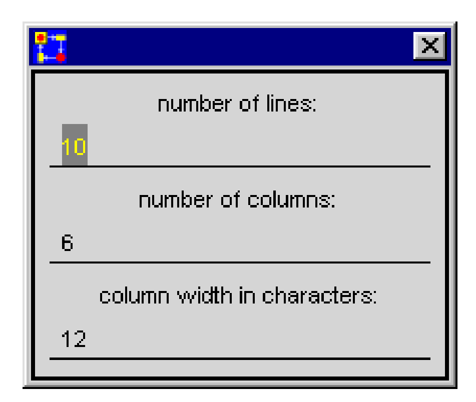


Abb. 6-39: Tabellen-Abfragefenster

Die einzelnen Zeilen des Fenster werden mit der Maus oder mit der Tab-Taste der Tastatur weitergeschaltet; die Return-Taste schließt

das Abfragefenster und zeigt ein Fenster mit der gewünschten Tabelle an. Diesem kann über die mittlere Maustaste und der dann verfügbaren Fenster-Menüfunktion ‘relable as...’ ein bestimmter Name zugewiesen werden, der von Inskriptionen her mit der ‘named:’-Funktion (siehe den nächsten Abschnitt) referenziert wird.

6.12.1 Eigenschaften und manuelle Bedienung

Wie Abb. 6-38 zeigt, besteht eine Tabelle aus in Zeilen und Spalten angeordneten Feldern, die als Druckknöpfe ausgebildet sind. Jedes Feld kann eine Text-Zeile aufnehmen. Die Maximallänge eines Feldes (Spaltenbreite) kann vom Benutzer für die ganze Tabelle oder spaltenweise festgelegt werden. Über jeder Spalte wird ein Spalten-Bezeichner angezeigt. Damit bei großen Tabellen nur der jeweils in Bearbeitung befindliche Teil angezeigt werden muß, sind sowohl die Spalten als auch die Zeilen mit jeweils einem Scroll-Balken versehen.

Damit ein Feld bearbeitet werden kann, muss es aktiviert werden. Dazu wird das Feld „angeklickt”, wodurch die Feldgrenzen invertiert werden; der Druckknopf ist „eingeschaltet“. In einem eingeschalteten Feld ist ein Text-Cursor sichtbar, über den das Feld editiert werden kann. Beim Editieren genügt es, daß sich der Maus-Cursor innerhalb der Tabelle befindet.

Nach Beenden des Editieren für ein Feld und Fortsetzen des Editiervorgangs in einem anderen Feld kann wie folgt verfahren werden:

- Anklicken des nächsten zu bearbeitenden Feldes.
- Drücken der Return-Taste wählt als nächstes Feld das erste Feld der nächsten Zeile aus. Falls es keine nächste Zeile gibt, wird das erste Feld der ersten Zeile aktiviert.
- Drücken der Tab-Taste aktiviert das nächste Feld in der gleichen Zeile. Gibt es kein nächstes Feld in der Zeile, so wird das erste Feld der nächsten Zeile ausgewählt. Gibt es keine nächste Zeile, so wird das erste Tabellenfeld aktiviert.

Soll kein weiteres Feld mehr bearbeitet werden, so wird enweder die rechte Maustaste gedrückt und die Funktion ‘accept’ ausgeführt oder man klickt mit der linken Maustaste auf eine Stelle, die nicht vion einem Feld belegt ist (z.B. zwischen oder unter die Felder). Der Inhalt des Feldes wird gespeichert und das Feld wird wieder normal dargestellt (Druckknopf erhaben). Die auf die gleiche Weise zugängliche Funktion ‘cancel’ kann während des Editierens verwendet werden, wenn die Feldänderungen rückgängig gemacht werden sollen.

6.12.2 Zuordnen einer Tabelle

Mit der folgenden Botschaft kann ein bestimmtes Fenster, in dem eine Tabelle dargestellt ist, angesprochen werden:

aTable := Table named: 'name'

Darin ist ‘name’ der Name des Fensters, in dem die Tabelle dargestellt ist. Beim Senden der Botschaft muß das Fenster bereits geöffnet sein.

6.12.3 Botschaften an Tabellen

6.12.3.1 Zugriff auf einzelne Zellen

Lesen und Schreiben einzelner Elemente einer Tabelle wird mit den Kommandos:

aTable cellAt: aCell

und

aTable cellAt: aCell put: aString

bewerkstelligt. aCell hat die gleiche Syntax wie ein Punkt: x@y.

Die Elemente einer Tabelle enthalten grundsätzlich nur Zeichenketten (Strings). Der Benutzer muß eine ggf. durchzuführende Wandlung vor der Ausgabe bzw. nach dem Einlesen eines Tabellenwertes selbst durchführen.

Beispiel:

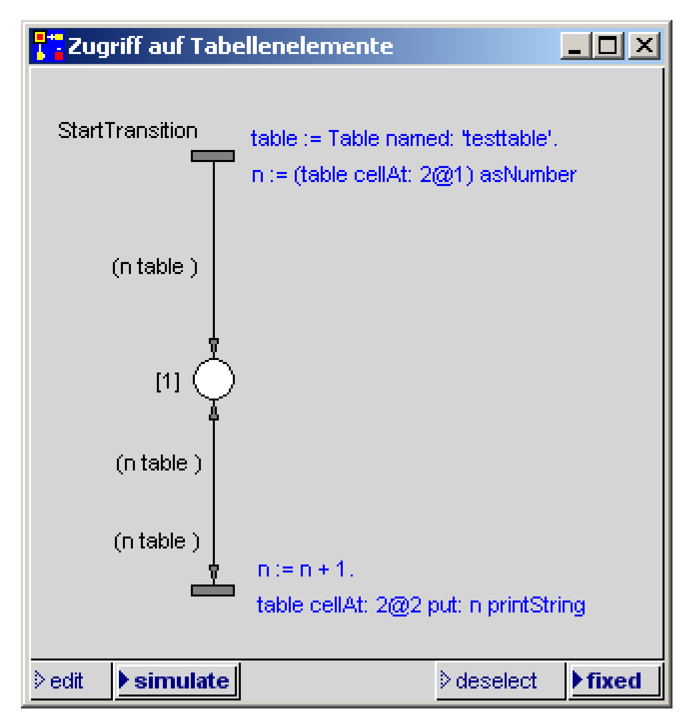


Abb. 6-40: Beispiel für die Ein/Ausgabe von/in Tabellenelemente

Abb. 6-40 zeigt ein einfaches Beispiel für die Eingabe von bzw. Ausgabe in Tabellen. Die Transition 'StartTransition' erzeugt nach dem Start eine Marke und dann keine weitere mehr, weil die Stelle nur maximal eine Marke aufnehmen kann. Aus dem Element 2@1 der Tabelle 'testtable' wird eine Zeichenkette eingelesen und in eine Zahl gewandelt. Die Tabellenidentifikation 'table' und der eingelesene Wert werden als Attribute an die Marke angeheftet und zu Stelle transportiert.

Danach wandert die Marke zwischen der Stelle und der unteren Transition hin- und her, wobei der Wert von n bei jedem Passieren der Transition erhöht und in das Element 2@2 der Tabelle ausgegeben wird.

6.12.3.2 Tabellen-Layout und Beschriftung

Mit der Botschaft:

aTable labelAt: spaltennummer

wird die Beschriftung der Spalte als String ausgelesen.

Die Spaltenbreite einer Spalte wird von der Botschaft:

aTable columnWidthAt: spaltennummer

als Ganzzahl abgeliefert. Sie gibt die Zahl der Zeichen an, die in der Spalte Platz finden.

Schließlich kann die Größen der Tabelle mit der Botschaft:

aTable dimensions

bestimmt werden. Sie wird nach der Syntax: spaltenzahl@zeilenzahl übergeben.

6.12.4 Tabellen-Menü

Das Tabellen-Menü ist in Abb. 6-38 abgebildet und enthält folgende Menüfunktionen:

edit cell...
Öffnet ein Fenster, in dem die Koordinaten des zu ändernden Elements anzugeben sind. Danach wird das Element invers dargestellt und, falls die Tabelle nur teilweise angezeigt wird, in den sichtbaren Bereich gebracht.

Die Funktion wird verwendet, wenn die Zuordnung zwischen Koordinaten und Element visuell schwierig ist oder sich das Element nicht im sichbaren Bereich der Darstellung befindet. Ansonsten ist das Anklicken des Elements einfacher und schneller.

go top
Bringt den Tabellenanfang in den sichtbaren Bereich.

go bottom
Bringt den Anfang der letzten Zeile der Tabelle in den sichtbaren Bereich.

insert row
Die Funktion öffnet ein Fenster, in dem die Anzahl der einzusetzenden Zeilen und die Zeile, nach der die neuen Zeilen einzusetzen sind, anzugeben ist.

remove row
Die Funktion öffnet ein Fenster, in dem die Anzahl der zu löschenden Zeilen und die Zeile, nach der die Zeilen gelöscht werden sollen, anzugeben ist.

column titles
In dem sich öffnenden Fenster ist eine Spaltennummer und der Spaltenbezeichner anzugeben.

display option
Die Funktion bearbeitet einen Schalter, der den Wert 'on' oder 'off' annehmen kann. Wird die Schalterstellung 'on' gewählt, so werden Änderungen in der Tabelle auch im Background-Simulationsmodus sofort dargestellt.



Abb. 6-41:Auswahlfenster

Bei Auswahl der Option 'off' wird der aktuelle Stand der Tabelle erst beim Anhalten der Simulation oder am Ende des Simulationslaufs ausgegeben. Wenn der Tabelleninhalt während des Ablaufs der Simulation nicht interessiert, empfiehlt sich die Option 'off', um die Simulation zu beschleunigen.

general column width
setzt die Spaltenbreite (Anzahl der Zeichen, die eine Spalte aufnehmen soll) aller Spalten der Tabelle auf den anzugeben Wert.

individual column width
In dem sich öffnenden Fenster ist eine Spaltennummer und die Breite für diese Spalte anzugeben. Alle anderen Spalten bleiben ungeändert.

text style
Definiert verschiedene Fonts für Zeichenausgabe in die Tabelle.

store
restore
siehe Abschnitt 6.2.

6.13 Allgemeine Histogramme

Histogramme (Balkendiagramme) werden für die Visualisierung der Häufigkeiten von verteilten Werten eingesetzt. Aus dem Histogramm kann man erkennen, wie sich Werte bzgl. eines vorgegebenen quantitativen Merkmals, dem die x-Achse zugeordnet wird, verteilen.

Beispiel: Abweichung eines Produktparameters vom Nennmaß.

Zeitabhängige Histogramme können in einfacher Weise direkt an Stellen und Konnektoren angehängt werden und sind im Kapitel: 'Der Simulator' beschrieben.

6.13.1 Arten von Histogrammen

Zur Darstellung der Häufigkeiten zerlegt man einen Bereich der x-Achse, in dem man alle oder die meisten Werte des betrachteten Merkmals erwartet, in n gleich lange Intervalle der Länge d. Soll ein Wert hinzugefügt werden, so wird er dem Intervall zugeordnet, in dem der Wert liegt. Über jedem Intervall wird ein Balken gezeichnet.

Die Höhe des Balkens hängt von der Art des Histogramms ab.

6.13.1.1 Standard-Histogramm

Es handelt sich hier um das Histogramm, welches normalerweise in den Lehrbüchern über Statistik als 'Histogramm einer Stichprobe' beschrieben wird.

Bezeichnen wir mit m(i,d), i=1,...,n die Anzahl der Werte, die im i-ten Intervall der Länge d liegen und mit m die Gesamtzahl der Werte, so wird über jedem Intervall (Klasse) ein Rechteck der Höhe

$$m(i,d) / m$$

errichtet (relative Klassenhäufigkeit). m(i,d) / m ist die Wahrscheinlichkeit dafür, daß ein Wert ins i-te Intervall fällt.

Das in Abb. 6-42 dargestellte Standard-Histogramm wurde in einem Workspace mit der Anweisungsfolge:

```
|histo verteil|
verteil := Normal mean: 5 deviation: 1.
histo := StandardHistogram named: 'Standard Histogram'.
1 to: 5000 do: [:i| histo addValue: verteil next].
```

erzeugt.

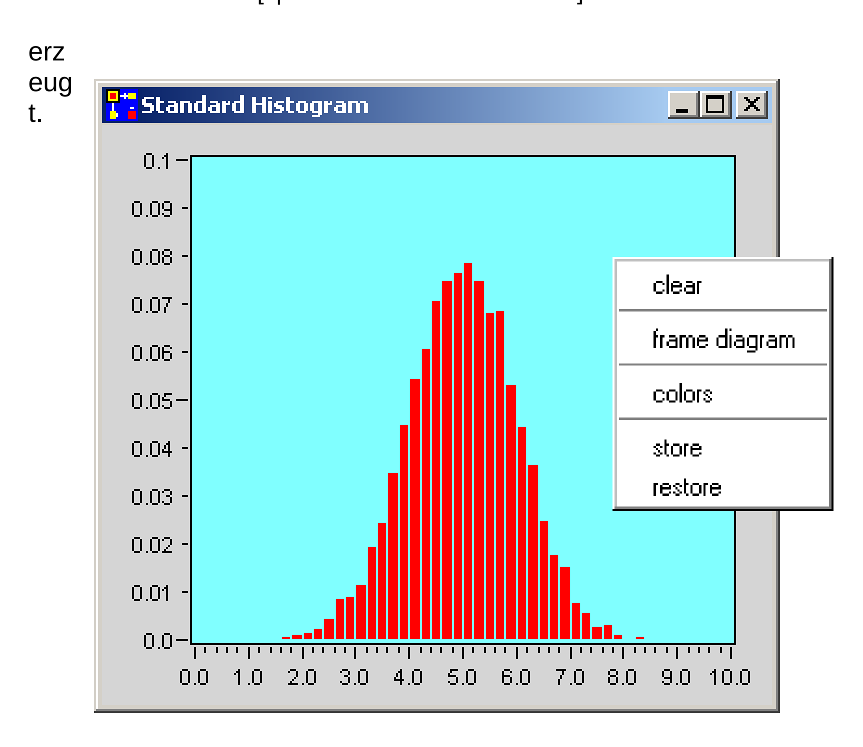


Abb. 6-42: Beispiel für ein Standard-Histogramm

6.13.1.2 Count-Histogramm

Über jedem Intervall wird ein Rechteck errichtet, dessen Höhe die Anzahl der Werte ist, die in dem Intervall liegen.

Das in Abb. 6-43 dargestellte Count-Histogramm wurde in einem Workspace mit der Anweisungsfolge:

```
|histo verteil|
verteil := Normal mean: 5 deviation: 1.
histo := CountHistogram named: 'Count Histogram'.
1 to: 5000 do: [:i| histo addValue: verteil next].
```

erzeugt.

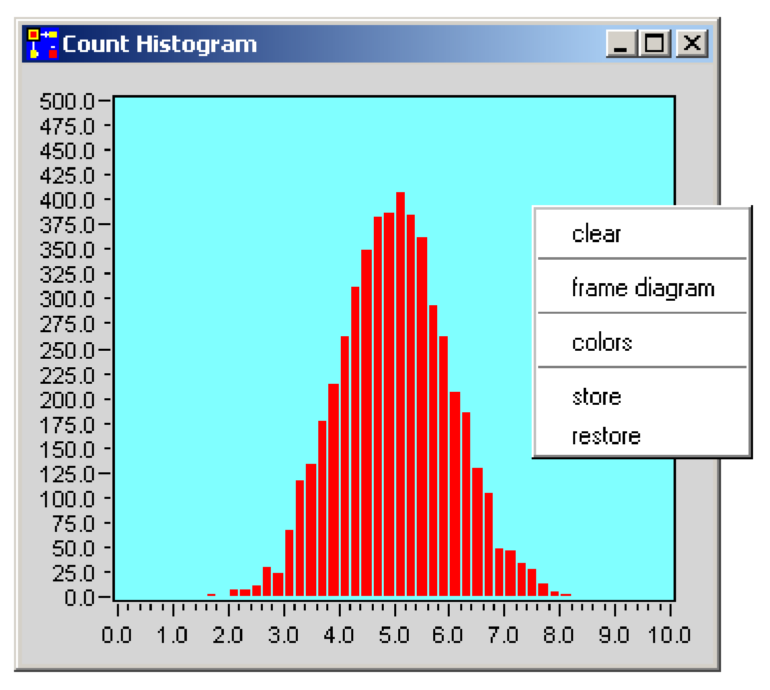


Abb. 6-43:Beispiel für ein Count-Histogram

6.13.1.3 Value-Histogramm

Über jedem Intervall wird ein Rechteck errichtet, dessen Höhe die Summe der Werte ist, die in dem Intervall liegen.

Das in Abb. 6-44 dargestellte Value-Histogramm wurde in einem Workspace mit der Anweisungsfolge:

```
|histo verteil|
verteil := Normal mean: 5 deviation: 1.
histo := ValueHistogram named: 'Value Histogram'.
1 to: 5000 do: [:i| histo addValue: verteil next].
```

erzeugt.

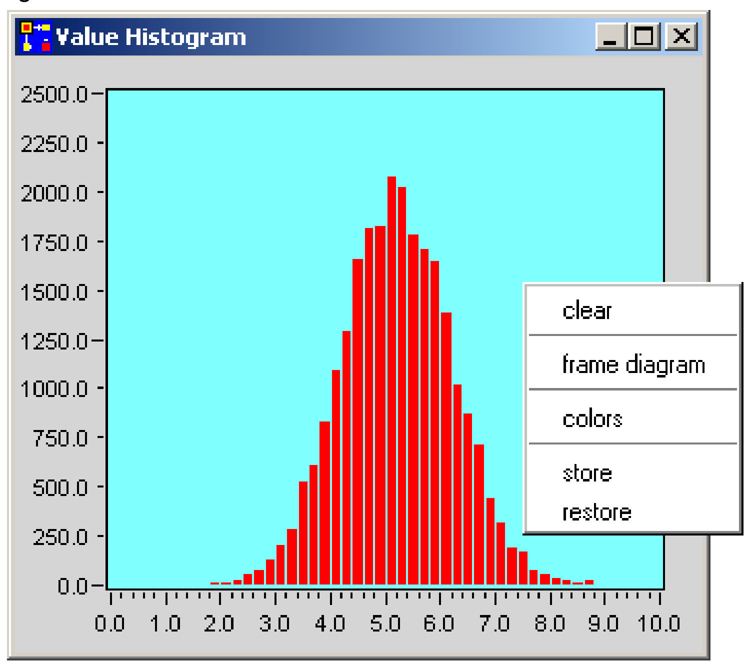


Abb. 6-44: Beispiel für ein Value-Histogramm

6.13.1.4 Distribution-Histogramm

Das Histogramm liefert eine Näherung für die Verteilungsfunktion und wird bei der Erstellung der „empirischen“ Wahrscheinlichkeits-verteilungen von PACE verwendet..

Bezeichnen wir mit m(i,d), i=1,...,n die Anzahl der Werte, die im i-ten Intervall der Länge d liegen und mit m die Gesamtzahl der Werte, so wird über jedem Intervall ein Rechteck der Höhe

$$m(i,d) / (m * d)$$

errichtet. Die Summe der Balkenflächen ist 1.

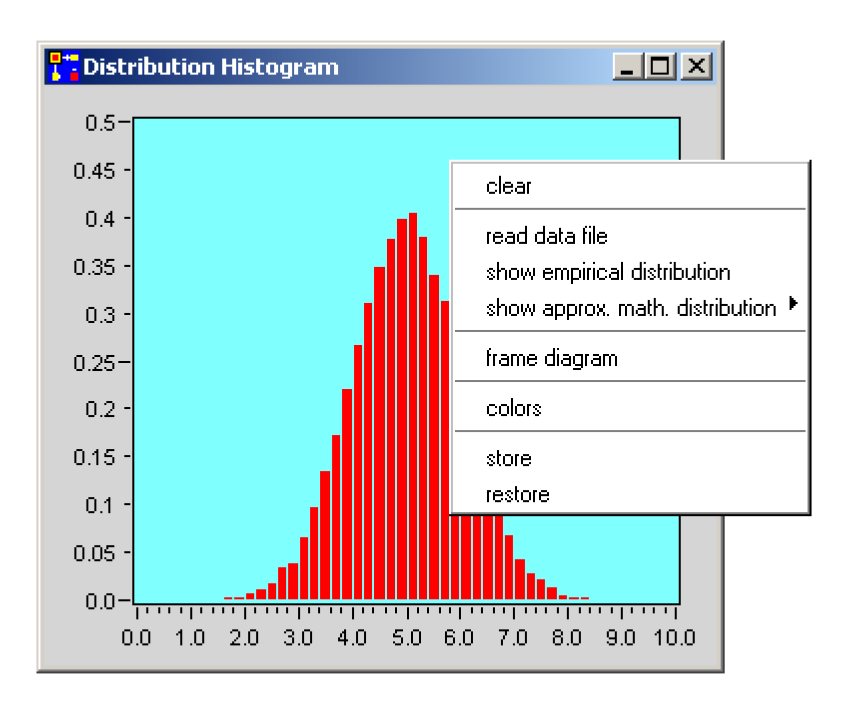


Abb. 6-45:Beispiel für ein Distribution-Histogramm

Die y-Werte des 'distribution histograms' stellen eine Näherung für die Verteilungsfunktion dar. Für d = 1 stimmt das 'distribution histogram' mit dem 'standard histogram' überein.

Das in Abb. 6-45 dargestellte Distribution-Histogramm wurde in einem Workspace mit der Anweisungsfolge:

```
|histo verteil|
verteil := Normal mean: 5 deviation: 1.
histo := DistributionHistogram named: 'Distribution Histogram'.
1 to: 20000 do: [:i| histo addValue: verteil next].
```

erzeugt.

6.13.2 Erzeugen und Zuordnen eines allgemeinen Histogramms

Eine Histogramm wird durch Ausführen einer der Menüpunkte der Funktion 'histograms' im 'view'-Menü der PACE-Hauptleiste erzeugt.

Mit einer der folgenden Botschaften kann ein bestimmtes Fenster, in dem ein allgemeines Histogramm eines bestimmten Typs dargestellt ist, an ein Modell angeschlossen werden:

aHistogram := StandardHistogram named: 'name'

aHistogram := CountHistogram named: 'name'

aHistogram := ValueHistogram named: 'name'

aHistogram := DistributionHistogram named: 'name'

name ist der Fenstername.

Beim Senden dieser Botschaft muß das Histogrammfenster bereits existieren.

6.13.3 Botschaften an Histogramme

Histogramme verstehen die folgenden beiden Botschaften:

addValue:
Mit dieser Botschaft kann ein Wert zu einem Histogramm hinzugefügt werden. Da sich die Anzahl der Werte verändert hat,

werden beim Standard- und beim Distribution-Histogramm alle Balken neu gezeichnet.

Beispiel:

aHistogram addValue: 5.6

clear
Das Histogramm wird zurückgesetzt, d.h. die Gesamtzahl und die Werte für die verschiedenen Intervalle werden auf 0 gesetzt.

Beispiel:

aHistogram clear.

mean
Liefert den Mittelwert des Histigramms.

standardDeviation
Liefert die Standard-Abweichung des Histigramms.

variance
Liefert die Varianz des Histogramms.

6.13.4 Achsen-Befehle

Diese sind im Kapitel 'Der Simulator', Abschnitt 'Zeitabhängige Diagramme' beschrieben.

6.13.5 Histogramm-Menüfunktionen

In den Abbildungen 6-42 bis 6-45 wurden die sog. Histogramm-Menüs angezeigt. Sie erscheinen, wenn der Cursor in den eingerahmten Teil eines Histogrammfensters platziert wird und danach die rechte Maustaste gedrückt wird. Die Histogramm-Menüs enthalten die gleichen Menüfunktionen; das Distribution-Histogramm

verfügt über zwei weitere Menüfunktionen, die für das Erzeugen von empirischen Verteilungen benötigt werden.

clear
Setzt alle Werte zurück. Mit dieser Funktion werden die Daten, die im Diagramm-Fenster dargestellt sind, gelöscht.

frame diagram
Definiert den Fenster-Ausschnitt, in dem das Diagramm dargestellt werden soll. Diese Funktion kann beispielsweise benutzt werden, wenn gewisse Einträge der Maßstab-Beschriftungen (Vermaßungen oder sonstiges) nicht gelesen werden können.

colors
Auswahl der Vorder- und Hintergrund-Farbe für das Histogrammfenster (siehe Abb. 6.3).

store
Speichert die Parameter und die Daten des Diagramm-Fensters in einem File mit frei wählbarem Namen. Das File wird im Unterverzeichnis 'ioutils' abgelegt und erhält eine der folgende Erweiterungen:

'.shst'	StandardHistogramm
'.chst'	CountHistogramm
'.vhst'	ValueHistogramm
'.dhst'	DistributionHistogramm

restore
Stellt die Parameter und die Daten eines Diagramm-Fensters wieder her. Dabei werden die Daten aus einem File mit einer der unter 'store' angegebenen Erweiterungen entnommen, das sich im Unter-Verzeichnis 'ioutils' befindet.

read data file
show empirical distribution
Diese Funktionen werden für die Erzeugung von empirischen Verteilungen benötigt. Sie sind zusammen mit ihrer Verwendung bei der

Erzeugung empirischer Verteilungen in Abschnitt 10.2.11 beschrieben.

show approx. math. distribution
Mit dieser Menüfunktion kann man visuell feststellen, ob sich die in dem Histogramm dargestellte Verteilung durch eine der folgenden mathematischen Verteilung approximieren lässt:

Abb. 6-46: Darstellung als mathematische Verteilung

Beta
Cauchy
Chi Square
Erlang
Exponential
Fisher Tippett
Gamma
Laplace
Logistical
Lognormal
Normal
Pearson Type 5

Bei Aufruf einer der Verteilungen öffnet sich ein Fenster in dem die gwählte mathematische Verteilung mit denselben charakteristischen Parametern wie das Histogramm (Mittelwert, Standard-Abweichung, Varianz) dargestellt wird.

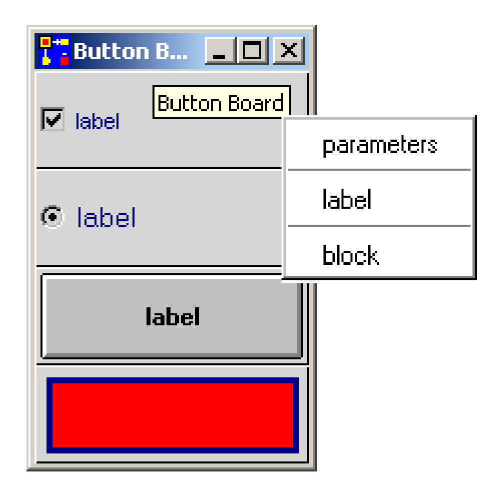
6.14 Knopfleiste (button board)

Knopfleisten werden für das Setzen von Flags, mit denen der Netzablauf beeinflußt werden kann, und für das getriggerte Ausführen von Smalltalk-Code verwendet. Flags können durch Drücken des Knopfs (mit der linken Maustaste) und durch Smalltalk-Botschaften gesetzt und zurückgesetzt werden. Jedem Knopf läßt sich außerdem ein Triggercode zuordnen, der bei Drücken des Knopfs ausgeführt wird.

Die folgenden Knopf-Layouts sind in PACE vorgesehen (Reihenfolge entspricht der in Abb. 6-47 abgebildeten vertikalen Knopfleiste):

- check box
- radio button
- trigger button
- color button.

Abb. 6-47:Knopftypen in einer Knopfleiste



Für den Zugriff auf die Knöpfe einer Leiste gibt es zwei Möglichkeiten, nämlich über die Eigenschaften eines Knopfs oder über die Position in der Leiste. Die erste Möglichkeit empfiehlt sich, wenn nur gelegentlich vom Programm auf die Knöpfe zugegriffen wird und die Eigenschaften eines Knopfs den Knopf eindeutig identifizieren. Andernfalls ist der Zugriff über die Position in der Leiste vorzuziehen.

6.14.1 Erzeugen und Zuordnen einer Knopfleiste

Eine Knopfleisten-Fenster wird durch Ausführen der Funktion 'button board' im 'view'-Menü der PACE-Hauptleiste erzeugt.

Beim Ausführen der Menüfunktion öffnet sich ein Fenster, in dem die Anzahl der Knöpfe zwischen 1 und 16 und die Anzahl der Zeilen, in denen die Knöpfe angeordnet werden sollen, angefordert wird. Durch geeignete Wahl dieser beiden Parameter kann man die Anordnung der Knöpfe (vertikale Leiste, horizontale Leiste, quadratischer Bereich, usw.) festlegen.

Mit der folgenden Botschaft kann ein bestimmtes Fenster, in dem eine Knopfleiste dargestellt ist, einem Netz zugeordnet werden:

aButtonBoard := ButtonBoard named: 'name'

Beim Senden dieser Botschaft muß die Knopfleiste (im Beispiel 'name') bereits existieren.

6.14.2 Zugriff über Merkmale

Das Setzen und Zurücksetzen eines Knopfs kann mit der linken Maustaste und durch Smalltalk-Botschaften an die Knopfleiste vorgenommen werden. Beim Umsetzen des Wertes eines Knopfs wird auch ein ggf. vorhandener Triggercode (siehe Knopf-Menü) ausgeführt.

Die hier angegebenen Botschaften greifen auf einen Knopf über dessen Eigenschaften (Name, Farbe, usw.) zu.

atColor:
Mit der Botschaft kann der aktuelle Wert eines Color-Button abgefragt werden:

aButtonBoard atColor: color

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. color ist ein String, der den Namen einer zulässigen Farbe enthält (z.B. 'red'), die in der angesprochenen Knopfleiste vorkommt. Die zulässigen Farben sind z.B. im Fenster 'Default Platform Colors' angegeben, das man über die Menüpunkte: 'View -> colors -> show default platform colors' der PACE-Hauptleiste anzeigen kann.

Die Anweisung liefert einen der Werte true oder false, je nachdem ob der Knopf gesetzt oder zurückgesetzt ist.

atColor:put:
Mit der Botschaft kann der aktuelle Wert eines Color-Button gesetzt oder zurückgesetzt werden:

aButtonBoard atColor: color put: aBoolean

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. color ist ein String, der den Namen einer zulässigen Farbe enthält (z.B. 'red'), die in der angesprochenen Knopfleiste vorkommt. Die zulässigen Farben sind z.B. im Fenster 'Default Platform Colors' angegeben, das man über die Menüpunkte: 'View -> colors -> show default platform colors' der PACE-Hauptleiste anzeigen kann. aBoolean kann nur die Werte true oder false annehmen.

atLabel:
Mit der Botschaft kann der aktuelle Wert eines Knopfs abgefragt werden:

aButtonBoard atLabel: label

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. label ist ein String, der den Namen des abzufragenden Knopfs angibt.

Die Anweisung liefert einen der Werte true oder false, je nachdem ob der Knopf gesetzt onder zurückgesetzt ist.

atLabel:put:
Mit der Botschaft kann der aktuelle Wert eines Button gesetzt oder zurückgesetzt werden:

aButtonBoard atLabel: label put: aBoolean

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. button ist ein String, der den Namen eines Knopfs in der Leiste angibt. aBoolean darf nur die Werte true oder false annehmen.

toggleAtColor:
Mit der Botschaft kann der aktuelle Wert eines Color-Button invertiert werden:

aButtonBoard toggleAtColor: color

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. color ist ein String, der den Namen einer zulässigen Farbe enthält (z.B. 'red'), die in der angesprochenen Knopfleiste vorkommt. Die zulässigen Farben sind z.B. im Fenster 'Default Platform Colors' angegeben, das man über die Menüpunkte: 'View -> colors -> show default platform colors' der PACE-Hauptleiste anzeigen kann.

toggleAtLabel:
Mit der Botschaft kann der aktuelle Wert eines Button invertiert werden:

aButtonBoard toggleAtLabel: label

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. label ist ein String, der den Namen des Knopfs angibt, dessen Flag invertiert werden soll.

triggerAtColor:
Mit der Botschaft kann der Code eines Color-Button ausgeführt werden (das Flag wird auf true gesetzt):

aButtonBoard triggerAtColor: color

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. color ist ein String, der den Namen einer zulässigen Farbe enthält (z.B. 'red'), die in der angesprochenen Knopfleiste vorkommt. Die zulässigen Farben sind z.B. im Fenster 'Default Platform Colors' angegeben, das man über die Menüpunkte: 'View -> colors -> show default platform colors' der PACE-Hauptleiste anzeigen kann.

triggerAtLabel:
Mit der Botschaft kann der Code eines Button ausgeführt werden (das Flag wird nicht verändert):

aButtonBoard triggerAtLabel: label

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. label ist ein String, der den Namen des Knopfs angibt, dessen Code ausgeführt werden soll.

6.14.3 Zugriff über Identifikation

Das Setzen und Zurücksetzen eines Knopfs kann mit der linken Maustaste und durch Smalltalk-Botschaften an die Knopfleiste vorgenommen werden. Beim Umsetzen des Wertes eines Knopfs wird auch ein ggf. vorhandener Triggercode (siehe Knopf-Menü) ausgeführt.

Die hier angegebenen Botschaften greifen auf einen Knopf über dessen Identifikation in der Knopfleiste zu.

getButtonColor:
Mit der Botschaft kann die Farbe eines Color-Button abgefragt werden:

aButtonBoard getButtonColor: aButton

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. aButton ist das Ergebnis der Methode getButton: oder ein Element des Arrays, den die Methode getAllButtons liefert.

Die Methode liefert die Farbe als Smalltalk-Symbol (z.B. #red).

getButtonFlag:
Mit der Botschaft kann der aktuelle Wert des Button abgefragt werden:

aButtonBoard getButtonFlag: aButton

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. aButton ist das Ergebnis der Methode getButton: oder ein Element des Arrays, den die Methode getAllButtons liefert.

Die Methode liefert einen der boolschen Werte true oder false, je nachdem ob der Button gesetzt oder nicht gesetzt ist.

getButtonLabel:
Mit der Botschaft kann der aktuelle Name des Button abgefragt werden:

aButtonBoard getButtonLabel: aButton

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. aButton ist das Ergebnis der Methode getButton: oder ein Element des Arrays, den die Methode getAllButtons liefert.

Die Methode liefert den Knopfnamen als String.

button:setColor:
Mit der Botschaft kann die aktuelle Farbe eines Color-Button gesetzt werden:

aButtonBoard button: aButton setColor: aColor

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. aButton ist das Ergebnis der Methode getButton: oder ein Element des Arrays, den die Methode getAllButtons liefert.

aColor ist ein Smalltalk-Symbol, das die gewünschte Farbe angibt (z.B. #green).

button:setFlag:
Mit der Botschaft kann der aktuelle Wert eines Button gesetzt oder zurückgesetzt werden:

aButtonBoard button: aButton setFlag: aBoolean

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. aButton ist das Ergebnis der Methode getButton: oder ein Element des Arrays, den die Methode getAllButtons liefert.

aBoolean darf nur die Werte true oder false annehmen.
button:setLabel:
Mit der Botschaft kann der Name eines Button festgelegt werden:

aButtonBoard button: aButton setLabel: aLabel

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. aButton ist das Ergebnis der Methode getButton: oder ein Element des Arrays, den die Methode getAllButtons liefert.

aLabel ist ein String, der den neuen Knopfnamen angibt (z.B. 'Knopf1').

getAllButtons
Die Methode liefert die Identifikationen aller Knöpfe einer Knopfleiste als Elemente eines Arrays.

Beispiel:

```
| buttons board|
board := ButtonBoard named: 'Test'.
buttons := board getAllButtons.
buttons do: [:but| board button: but setLabel: 'krr'].
buttons do: [:but| board button: but setFlag: true].
buttons do: [:but| board button: but setColor: #green].
```

getButton:
Die Methode

aButtonBoard getButton: aLabel

liefert die Identifikation eines Knopfes, über den der Knopf in anderen Methoden angesprochen werden kann.

Beispiel:

```
| knopf1 board|
board := ButtonBoard named: 'Test'.
knopf1 := board getButton: 'go'.
board button: knopf1 setColor: #blue.
```

toggleButton:
Die Methode invertiert den aktuellen Wert des Knopfes:

aButtonBoard toggleButton: aButton.

aButtonBoard ist das Resultat einer 'named:'-Botschaft an eine Knopfleiste. aButton ist das Ergebnis der Methode getButton: oder ein Element des Arrays, den die Methode getAllButtons liefert.

Beispiel:

Das folgende Codestück invertiert alle Knöpfe einer Knopfleiste 'Test':

```
| buttons board|
board := ButtonBoard named: 'Test'.
buttons := board getAllButtons.
buttons do: [:but| board toggleButton: but].
```

6.14.4 Knopf-Menü

Das Knopfmenü besteht aus drei Zeilen (siehe Abb. 6-47), denen die folgenden Menüfunktionen zugeordnet sind:

parameters
Mit dieser Menü-Funktion kann das Aussehen eines Knopfs (appearance) und seine Verarbeitungsfunktion (control function) festgelegt werden. In Abb. 6-48 sind die Vorbesetzungen eines Knopfs beim Erzeugen einer Knopfleiste angegeben.

Die möglichen Layouts sind in der linken Hälfte der Abbildung angegeben und wurden zu Beginn von Abschnitt 6.14 schon beschrieben. Bei drei der in Abb. 6-47 dargestellten Knöpfen ist jeweils das zugeordnete Flag gesetzt (Wert: true).

Ist der Color-Button nicht umrahmt, ist die 'check box' leer oder sind die übrigen Knöpfe schwach gezeichnet, so ist das zugeordnete Flag zurückgesetzt (Wert: false).

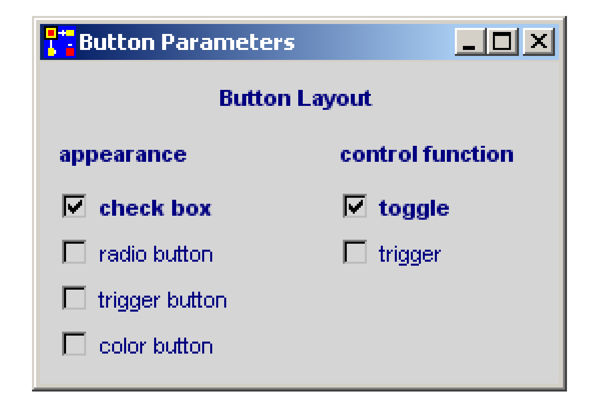


Abb. 6-48:Parameterfenster eines Knopfes

Bei Drücken des Knopfs mit der linken Maustaste invertiert die Verarbeitungsfunktion "toggle" das Flag des Knopfs und führt danach seinen Blockcode aus. Die Funktion kann deshalb auch zum Triggern verwendet werden. Die Funktion "trigger" führt den Blockcode bei Drücken des Knopfes mit der linken Maustaste aus und weist dem Knopfwert flag den Wert true zu.

label
dient zur Änderung eines Knopfnamens. Die Funktion öffnet ein Fenster, in das der neue Name eingetragen wird. Mit der Menüfunktion 'accept' oder der Return-Taste wird der neue Name von PACE übernommen.

block
gibt den Smalltalk-Code an, der beim Drücken eines Knopfs ausgeführt werden soll.

Der Knopfcode aller Knöpfe ist nach dem Erzeugen einer Knopfleiste leer (siehe Abb. 6-49). Bei der Formulierung des Knopfcodes stehen die gleichen Methoden zur Verfügung, die auch bei den 'Extra Codes' verwendet werden können.

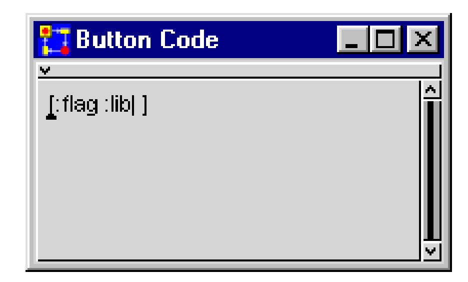


Abb. 6-49: Fenster für die Eingabe des Triggercodes eines Knopfes

Der Block besitzt zwei Parameter:

- flag — liefert den aktuellen Wert des flag.
- lib — liefert den Bezugspunkt für die Bibliothek von PACE-Methoden, welche die in diesem Kontext verwendbaren Smalltalk-Methoden enthält.

Beispiel:
Falls der Wert des flag = true ist, wird in die Stelle 'place' eine Marke gelegt.

```
[:flag :lib|
 flag ifTrue: [lib addTokenTo: (lib placeNamed: 'place')]]
```

7 NETZEDITOR

Mit dem PACE-Netzeditor (im folgenden auch kurz mit 'Editor' bezeichnet) können Modelle graphisch konstruiert, spezifiziert und verändert werden. Verschiedene Fenster, in denen jeweils ein Teilnetz des aktuellen PACE-Modells dargestellt ist, können gleichzeitig geöffnet und bearbeitet werden. Die Zahl der geöffneten Fenster ist nicht beschränkt.

Der graphische Editor überprüft die syntaktische Korrektheit eines PACE-Modelles während der Eingabe. Er wird für ein Netz-Fenster durch Betätigen des 'editor'-Schalters im unteren Bereich des Netz-Fensters aktiviert.

7.1 Einfügen neuer Elemente

7.1.1 Einfügen von S- und T-Elementen

Die S-Elemente (Stellen und Kanäle) und T-Elemente (Transitionen und Module), die während der Modellierung eingefügt werden, sind im Hauptmenü des Editors aufgelistet. Dieses Menü erscheint durch Drücken der rechten Maustaste, wenn sich der Cursor im Netz-Fenster befindet und kein Element markiert wurde (sog. No-Selection-Menü). Das im angezeigten Menü gewählte Element wird an der aktuellen Cursorposition angezeigt. Es kann mit der Maus frei bewegt und verschoben werden.

Durch Drücken der linken Maustaste wird die Position des Elementes eingefroren. Wird eine andere Taste gedrückt oder soll das Element außerhalb des zur Verfügung stehenden Arbeitsfeldes positioniert werden, so wird das Element wieder gelöscht.

7.1.2 Einfügen von Konnektoren

Die Konnektoren zwischen zwei Netz-Elementen werden direkt, ohne ein Menü erstellt. Dies geschieht folgendermaßen: Mit dem Cursor auf das Element fahren, von dem aus die Verbindung gezogen werden soll. Die linke Maustaste drücken und diese gedrückt halten. Auf dem Bildschirm erscheint an der Cursorposition ein Fadenkreuz an Stelle der üblichen Cursoranzeige. Mit dem Cursor auf das Ziel-Element fahren und die Taste wieder loslassen.

Durch Wiederholen desselben Vorganges in die entgegengesetzte Richtung, kann ein sogenannter "Doppelkonnektor" erstellt werden, der in beide Richtungen weist. Dieser ist die Zusammenfassung von zwei gegenläufigen Konnektoren.

7.1.3 Einfügen von Marken

In einem PACE-Modell können Marken sowohl bei der erstmaligen Spezifikation als auch zu einem beliebigen späteren Zeitpunkt während der Simulation auf eine Stelle gesetzt, wieder gelöscht oder durch Verändern ihrer Attribute neu definiert werden.

Im Editier-Modus wird der Initialzustand der Marken festgelegt. Im Simulations-Modus kann nur eine vorübergehende Änderung der Markierung durchgeführt werden. Diese Änderung wird dann nur bei der Fortsetzung der aktuellen Simulation berücksichtigt.

Eine Marke wird auf eine Stelle gesetzt, indem diese zunächst markiert oder angewählt wird (Cursor auf die Stelle positionieren und die linke Maustaste drücken). Danach wird aus dem Menü, das durch Drücken der rechten Maustaste erscheint, im Editier-Modus 'initial tokens', im Simulations-Modus 'tokens' ausgewählt.

Es erscheint ein Marken-Listenfenster, das benutzt wird, um die Marken einer Stelle neu zu setzen, zu initialisieren oder um deren

aktuelle Werte zu verändern. Die Marken-Fenster sind Listen-Fenster und werden in drei Unterfenster aufgeteilt.

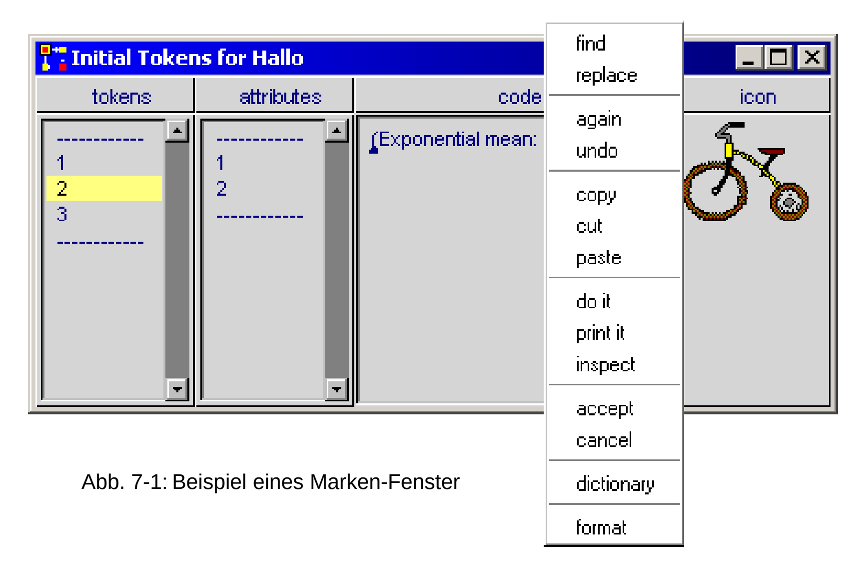


Abb. 7-1: Beispiel eines Marken-Fenster

Das linke Unterfenster zeigt eine Auflistung der Marken-Indizes. Es trägt die Überschrift 'tokens'. In diesem Unterfenster können bestehende Marken gelöscht und neue hinzugefügt werden. Das Auswahlmenü mit den entsprechenden Funktionen wird durch Drücken der rechten Maustaste angezeigt. Es wird weiter unter beschrieben.

Sobald eine Marke in der Liste mit der linken Maustaste angewählt wird, zeigt das mittlere Unterfenster eine Auflistung mit den Indizes der Attribute der gewählten Marke. Dieses mittlere Unterfenster trägt die Überschrift 'attributes'. Auch hier steht ein Menü zur Verfügung (rechte Maustaste), das weiter unten beschrieben wird. In diesem mittleren Unterfenster können bestehende Attribute der gewählten Marke durch Auswählen des entsprechenden Menüpunktes gelöscht oder neue hinzugefügt werden.

Das dritte Unterfenster ist ein Text-Fenster. Es trägt die Überschrift 'code'. Darin können die Inskriptionen der jeweiligen Markenattribute

angezeigt und geändert werden. Die Attribute sind Smalltalk-Codestücke, die als Teil der Funktion 'initialize' vor dem Start eines Simulationslaufs ausgeführt werden.

Das vierte, rechte Teilfenster zeigt, falls der Marke eine Ikone zugeordnet wurde, das Bild der Ikone. Diese Ikone ersetzt des Standardsymbol für die Marke auf dem Weg von der Stelle, in der die Initialmarke gesetzt wurde, bis zur ersten Transition.

Abbildung 7-1 zeigt ein Marken-Listenfenster mit drei Initialisierungs-Marken für die Stelle ‘stelle1’. Die zweite selektierte Marke hat zwei Attribute. Das erste Attribut ist der Code ‘Exponential mean: 10’. Das rechte Fenster zeigt die zugeordnete Ikone, ein Dreirrad.

7.1.3.1 Marken-Menüs

Es gibt zwei Marken-Menüs, je nachdem ob eine Zeile markiert wurde oder nicht. Die Funktionen werden im folgenden gemeinsam beschrieben. Folgende Funktionen stehen im Marken-Listenfenster zur Verfügung, wenn sich der Cursor im linken Unterfenster mit der Überschrift 'tokens' befindet. Diese Funktionen beeinflussen die Marke und erscheinen durch Drücken der rechten Maustaste.

add
Fügt am Ende der Liste eine neue Marke hinzu.

insert
Fügt vor der selektierten Marke eine neue hinzu.

paste
Nur für Initialisierungs-Marken verfügbar. Fügt den Inhalt des Paste-Buffers (eine Initialisierungs-Marke) zu der Liste der Initialisierung-Marken hinzu.

copy
Nur für Initialisierungs-Marken verfügbar. Kopiert die Initialisierungs-Marke in den Paste-Buffer.

remove

Löscht die angewählte Marke. Initialisierungs-Marken werden in den Paste-Buffer kopiert.

assign icon
Es öffnet sich ein Auswahlfenster, in dem die Namen aller verfügbaren Ikonen aufgelistet sind. Durch Anklicken eines Namens wird die Ikone der Marke zugeordnet und im rechten Teilfenster angezeigt.

remove icon
Der Marke wird das Standardsymbol für Marken (kleiner gefüllter Kreis) zugeordnet.

...
Unterdrückt das Anzeigen der Attribute aller auf der Stelle gespeicherten Marken im Netzfenster.

7.1.3.2 Attribute-Menüs

Folgende Befehle stehen im Marken-Listenfenster zur Verfügung, wenn sich der Cursor im mittleren Unterfenster mit der Überschrift 'attributes' befindet. Diese Befehle beeinflussen die Attribute der angewählten Marke und erscheinen durch Drücken der rechten Maustaste.

insert
Fügt vor dem angewählten Attribut ein neues hinzu.

remove
Löscht das angewählte Attribut.

add
Fügt am Ende der Liste ein neues Attribut ein.

inspect
Dieser Befehl ist nur im Simulations-Modus verfügbar. Damit wird ein Smalltalk-Inspector für das angewählte Marken-Attribut geöffnet.

7.1.3.3 Spezifizieren und Ändern der Attribute

Im rechten Unterfenster können für das angewählte Attribut beim erstmaligen Einfügen Werte spezifiziert bzw. bei bereits bestehenden Attributen die Werte geändert oder neue hinzugefügt werden. Diese Operationen werden erst dann wirksam, wenn der 'accept'-Befehl ausgeführt wurde, der eingegebene Smalltalk-Code also übernommen wurde.

Die Funktionen des mit der rechten Maus-Taste anwählbaren Auswahlmenüs sind in Abschnitt 4.5: 'Texteingabe' beschrieben.

7.2 Inskribieren von Elementen

Die Befehle, die für das Inskribieren von Elementen zur Verfügung stehen, können durch Drücken der rechten Maustaste aktiviert werden, wenn das entsprechende Element zuvor angewählt wurde.

7.2.1 Inskribieren von S- und T-Elementen

Text-Fenster für das Anbringen der Einträge
Ein Text-Fenster wird mit wählbarer Größe geöffnet, sobald die entsprechende Funktion angewählt wurde. Die Größe und die Anfangsposition wird beim Öffnen des Fensters festgelegt. Befindet sich der Cursor innerhalb des Fensters, dann kann mit der linken Maustaste (durch Überstreichen mit dem Cursor) Text markiert und für bestimmte Text-Manipulationen ausgewählt werden. Die rechte Maustaste gibt die Text-Bearbeitungsfunktionen frei (siehe Kapitel 'Benutzerschnittstelle'). Die Inskription wird durch Ausführen der 'accept'-Funktion vom Smalltalk-Compiler überprüft und, falls keine Fehler festgestellt werden, in das Element aufgenommen.

Verbergen von Code-Einträgen
Gelegentlich ist es nicht erwünscht, den inskribierten Smalltalk-Code einer Transition immer im Netz-Fenster anzuzeigen. Das kann z.B. der Fall sein, wenn einer Transition sehr viel Code zugeordnet wurde und dadurch die Darstellung eines Teilnetzes unübersichtlich wird, Ein anderer sehr wichtiger Fall liegt vor, wenn ein Anwender ein Modell nur verwenden, betrachten oder begutachten will, sich aber für die Details der Implementierung selbst nicht interessiert.

In diesem Fall wird die '...'-Funktion ("Drei-Punkte-Funktion") in einem 'condition'-, 'delay'- oder 'action'-Menü ausgeführt. Im Netzfenster wird danach nur ... angezeigt. Durch erneute Ausführung der Drei-Punkte-Funktion kann der Text im Netz-Fenster wieder sichtbar gemacht werden.

7.2.2 Konnektor-Attribute

Konnektoren können mit Attributen versehen werden. Dies geschieht in einem Fenster, das ähnlich wie ein Marken-Listenfenster aufgebaut ist. Es hat nur zwei Unterfenster, nämlich links das Attributfenster, das dem Unterfenster 'attributes' in Abb. 7-1 entspricht und rechts das Codefenster, welches das zugeordnete Attribut (ein Literal, d.h. eine Zahl, einen Bezeichner, usw.) aufnimmt (siehe dazu auch Abschnitt 7.1: ‘Einfügen neuer Elemente’).

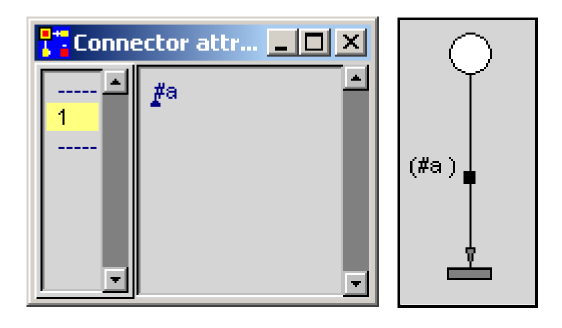


Abb. 7-2: Ein Konnektor mit Attribut-Menü

7.3 Netz-Editor-Menüs

Die Menüs, die in einem Editier-Fenster aufgerufen werden, sind von den jeweils angewählten Elementen abhängig. Ist das Editier-Fenster leer oder wurden keine Netz-Elemente angewählt, so erscheint bei Drücken der rechten Maustaste das Editor-Hauptmenü (das in Abb. 7-3 dargestellte Menü gilt für ein Unternetz; im Hauptnetz fehlt die Zeile ‘supernet’). Mit diesem Menü können einzelne Elemente (Stelle, Kanal, Transition, Modul) oder Kommentare und Inskriptionen in das zu editierende Netz eingetragen oder andere, das gesamte Netz betreffende Befehle, ausgeführt werden. Wurde ein Netz-Element angewählt, erscheint durch Drücken der rechten Maustaste ein Menü, das die Befehle auflistet, die für dieses Element zur Verfügung stehen. Es folgt eine Beschreibung aller Editor-Menüs.

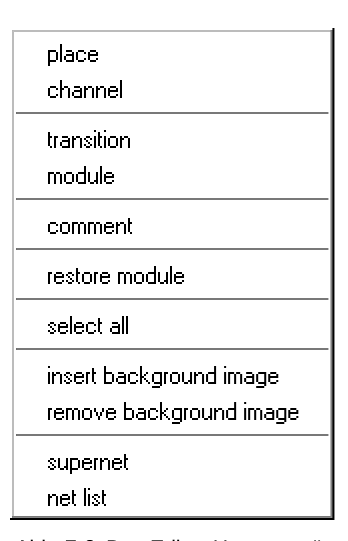


Abb. 7-3: Das Editor-Hauptmenü

Der Editor wird durch Betätigen des 'editor'-Schalters im unteren linken Bereich des Netz-Fensters aktiviert.

7.3.1 Editor-Hauptmenü

Eines der im folgenden angegebenen Elemente kann nach seiner Erzeugung durch Verschieben der Maus frei im Fenster bewegt und positioniert werden. Klicken mit der linken Maustaste fixiert das Element, durch Klicken mit der rechten Maustaste wird der Vorgang abgebrochen.

place
Erstellt eine neue Stelle. Die gewählte Standard-Ikone erscheint an der Cursorposition.

channel
Erstellt einen neuen Kanal.

transition
Erstellt eine neue Transition.

module
Erstellt einen neuen Modul.

comment
Unter Verwendung eines Textfensters wird eine neue Kommentar-Box erstellt. Zur Festlegung der Größe und Position der Kommentar-Box siehe Abschnitt 4.7.2: 'Festlegen der Fenstergröße beim Öffnen'. (Die Größe und Position der Kommentarbox kann später mit den Standard-Windows-Funktionen verändert werden.)

restore module
Zeigt ein Selektionsmenü (Abb. 7-4) mit den Namen aller im Default-Verzeichnis für Module gespeicherten Unternetze an. Durch Anwählen eines dieser Namen und Drücken des 'ok'-Knopfs, wird das entsprechende Unternetz mit seinen Interface-Stellen und Kanälen in das aktuelle Netz hineingeladen.

Bei der Integration wird das aktuelle Netz über Konnektoren mit den das Interface bildenden S-Elementen des geladenen Moduls verbunden. Die so entstandenen Schnittstellen werden in allen Ebenen des Hierarchie-Baumes des geladenen Netzes, in denen sie vorkommen,

ersetzt. Das neue Unternetz wird in die Netz-Liste aufgenommen und durch Einrücken in der entsprechenden Hierarchiestufe dargestellt.

select all
Wählt alle S- und T-Elemente im aktiven Fenster an (die Konnektoren werden nicht selektiert).

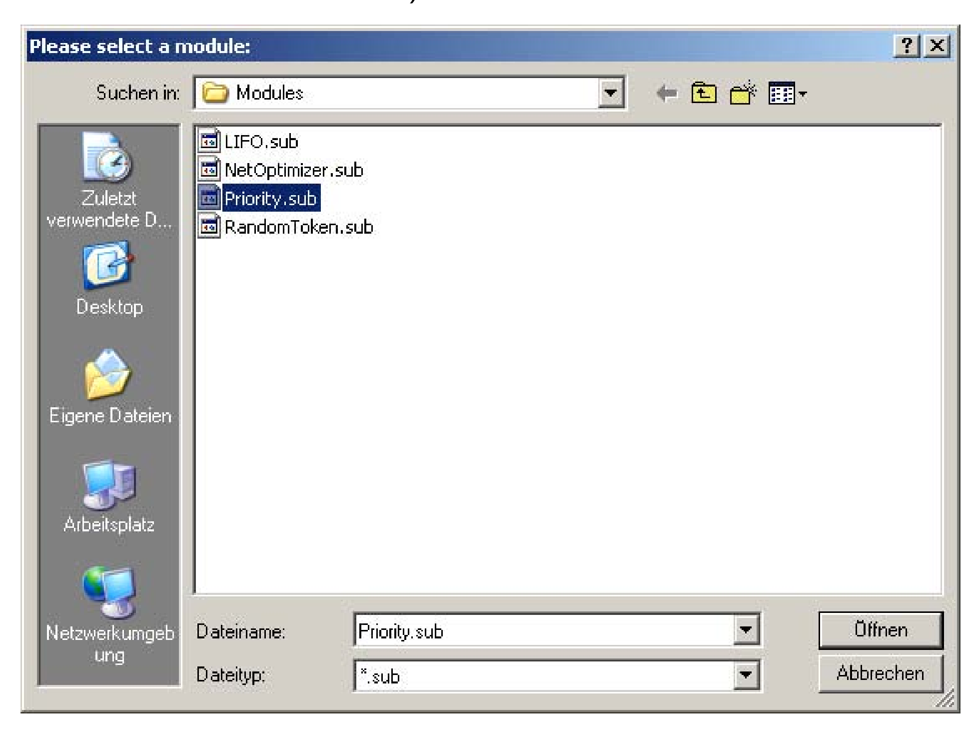


Abb. 7-4: 'restore module'-Selektionsmenü

insert background image
Es erscheint ein Auswahlmenü mit den verfügbaren Grafiken (Ikonen). Nach Auswahl einer Grafik wird diese als Hintergrundbild in das aktuelle Fenster eingepaßt.

Je nach Intensität der Ikone wird dabei die Sichtbarkeit des im Fenster dargestellten Netzes beeinträchtigt. Diese kann durch Bleichen der Ikone verbessert werden (siehe Abschnitt 5.10.2, Menüfunktion ‘fade’).

Bei Hintergrundbildern sind folgende Beschränkungen zu beachten:

- Während sich Netzfenster ohne Hintergrundbilder überdecken dürfen und die Restaurierung bei der Aktivierung eines verdeckten Netzfensters normalerweise kaum bemerkbar ist, sollten Hintergrundbilder nicht in Fenstern verwendet werden, die von anderen Netzfenstern verdeckt werden. Ihre Verwendung führt in diesem Fall bei häufiger Restaurierung zu einem Flackern des Fensters.

- In Netzfenstern mit Hintergrundbildern sollten die Ikonen für statische Netzelemente (Stelle, Kanal, Transition, Modul) nicht über Inskriptionen während des Simulationslaufs ausgetauscht werden. Auch in diesem Fall führt häufiges Austauschen einer Ikone zu einem Flackern des Fensters.

Die Verwendung von Hintergrundbildern ist bei der Entwicklung und beim Öffnen von Netzfenstern rechenintensiv. Beim Ausführen von Modellen und beim Laden von Images entsteht keine merkbare Verzögerung.

Günstige ist deshalb auf PC die Verwendung eines schnellen Pentium. Damit unnötige Wandlungen und damit die 'fade'-Methode einsetzbar ist, sollte die Graphikkarte eine Bildtiefe von 24 Bit ermöglichen.

Hintergrundbilder sollten erst bei den abschließenden Arbeiten an einem Modell, z.B. als letzter Modellierungsschritt vor der Auslieferung eines Anwendungs-Simulators, eingesetzt werden.

remove background image
Ein ggf. vorhandenes Hintergrundbild wird aus dem Fenster entfernt.

supernet
Dieser Menüpunkt wird nur angezeigt, wenn es ein in der Hierarchie höher liegendes Fenster gibt.

Das im Hierarchiebaum höher liegende Netz-Fenster wird angezeigt. Der Cursor wird auf die Schnittstellen-Ikone des Subnetzes gestellt, von dem aus diese Funktion aufgerufen wurde. Ist für das höherliegende Netz noch kein Fenster vorhanden, so wird ein Fenster erzeugt und geöffnet.

net list
Durch Ausführen dieses Befehles wird das Fenster mit der Netz-Liste in den Vordergrund gebracht. Der Cursor wird auf den Namen des Subnetzes gestellt, von dem aus diese Funktion aufgerufen wurde.

7.3.2 Stellen-Menü

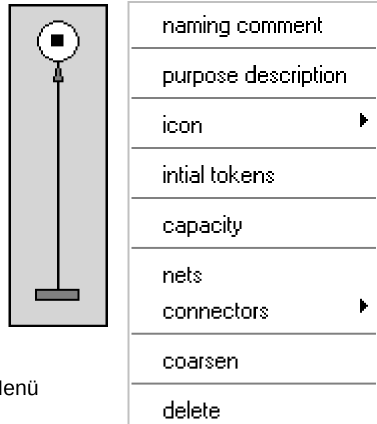


Abb. 7-5: Stellen-Menü

naming comment
Es wird ein Text-Fenster geöffnet, in dem der markierten Stelle ein benennender Kommentar zugeordnet werden kann, über den sie von Inskriptionen her angesprochen wird.

purpose description
Hier kann, falls im Einzelfall sinnvoll, eine Beschreibung an die Stelle angehängt werden, die ihren Sinn und Zweck angibt. Hierzu gehören gegebenenfalls der Grund, warum die Kapazität beschränkt wurde, warum welche ‘initial tokens’ eingeführt wurden, was die Kopplung mit anderen Netzelementen bewirkt, usw.

Die ‘purpose description’ kann dazu verwendet werden, um eine umgangssprachliche Beschreibung der Stelle und ihrer Aufgabe im Netz niederzulegen und damit auch die Entwurfsgedanken im Modell zu verankern. Diese können bei späteren Erweiterungen sehr wertvoll sein, insbesondere dann, wenn das Modell längere Zeit nicht bearbeitet wurde und/oder neue Mitarbeiter eingearbeitet werden sollen.

icon
Erlaubt die Wahl, die Definition und die Änderung der Ikone für die Stelle. Nach der Anwahl dieser Funktion erscheint ein Folgemenü, das weiter unten in diesem Kapitel beschrieben wird.

initial tokens
Beschreibt den Anfangszustand für die markierte Stelle. Dazu wird ein Fenster mit einer Auflistung der Marken, die beim Initialisieren für diese Stelle erzeugt werden, geöffnet. In diesem Fenster können sowohl neue Marken gesetzt als auch bestehende gelöscht oder verändert werden (siehe auch Abschnitt 7.1.3).

capacity
Beschränkt die Aufnahmekapazität der angewählten Stelle. Die Kapazität einer Stelle ist eine positive ganze Zahl, welche die Maximalzahl der Marken, die sich gleichzeitig auf dieser Stelle befinden dürfen, angibt. Die Eingabe des Wertes 0 (Null) bedeutet, daß beliebig viele Marken auf der Stelle liegen dürfen.

nets
Zeigt eine Liste aller Teil-Netze an, in denen die selektierte Stelle vorkommt. Durch Anwählen eines dieser Netze wird ein Fenster mit dem entsprechenden Teil-Netz geöffnet oder aktiviert. Sobald das

Netz-Fenster für das Teil-Netz geöffnet ist, springt der Cursor automatisch auf die selektierte Stelle.

connectors
Nach der Wahl dieses Menüpunktes erscheint ein Untermenü mit der Auflistung aller möglichen Konnektor-Arten. Nachdem eine Auswahl getroffen wurde, wird eine Liste aller Konnektoren des entsprechenden Typs zu dieser Stelle angezeigt. Das Anwählen eines Listenpunktes bewirkt, daß ein Fenster mit dem Netz, welches diesen Konnektor enthält, geöffnet oder aktiviert wird. Sobald das Netz-Fenster geöffnet ist, springt der Cursor automatisch auf die entsprechende Stelle.

coarsen
Es wird ein neuer Kanal erzeugt, der die angewählte Stelle enthält.

delete
Löscht nach der Bestätigung durch den Benutzer die Stelle und alle damit verbundenen Konnektoren. Dieser Befehl kann nur in dem Fenster ausgeführt werden, in dem die Stelle erzeugt wurde. In Unternetzen kann diese Stelle, die ja dort als Schnittstelle zum hierarchisch höher gelegenen Netz mit schwächer gezeichneter Ikone dargestellt ist, nicht gelöscht werden.

7.3.3 Kanal-Menü

naming comment
Um die Lesbarkeit einer PACE-Spezifikation zu erhöhen, kann mit dieser Funktion in einem Textfenster der Name des Kanals bzw. ein Kommentar zu dem Kanal erstellt oder geändert werden.

Der Kommentar/Name eines Kanals läßt sich an irgendeine Stelle im Netz-Fenster verschieben (siehe dazu Kapitel 9:'Verschieben von Text').

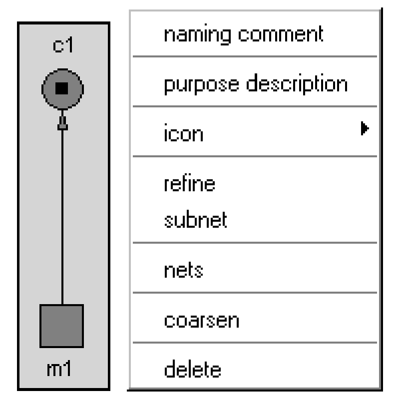


Abb. 7-6: Das Kanal-Menü

purpose description
Hier kann, falls im Einzelfall sinnvoll, eine Beschreibung an den Kanal angehängt werden, die seine Aufgabe im Netz angibt.

Die ‘purpose description’ kann dazu verwendet werden, um eine umgangssprachliche Beschreibung niederzulegen und damit auch die Entwurfsgedanken im Modell zu verankern. Diese können bei späteren Erweiterungen sehr wertvoll sein, insbesondere dann, wenn das Modell längere Zeit nicht bearbeitet wurde und/oder wenn neue Mitarbeiter eingearbeitet werden sollen.

icon

Erlaubt die Wahl, die Definition und die Änderung der Ikone für den Kanal. Nach der Anwahl dieser Funktion erscheint ein Folgemenü, das weiter unten beschrieben wird.

refine
Dieses ist die inverse Operation zu 'coarsen'. Der angewählte Kanal wird in seine einzelnen Elemente aufgelöst und in das hierarchisch höhere Netz integriert.

Bei der Ausführung dieser Funktion ist zu beachten, daß nicht mehr benötigte Konnektoren von S-Elementen zu Netzen automatisch eliminiert werden, weil keine Verbindung zu einer Transition in einem hierarchisch untergeordneten Netz mehr besteht.

subnet
Öffnet ein Netz-Fenster, in dem das Kanal-Netz dargestellt wird.

nets
Zeigt eine Liste aller (Unter-)Netze an, in denen der selektierte Kanal vorkommt. Durch Anwählen eines dieser Netze wird ein Fenster mit dem entsprechenden Netz geöffnet oder aktiviert. Sobald das Netz-Fenster geöffnet ist, springt der Cursor automatisch auf den entsprechenden Kanal.

coarsen
Es wird ein neuer Kanal erzeugt, der den angewählten Kanal enthält.

delete
Löscht nach der Bestätigung durch den Benutzer den Kanal und alle damit verbundenen Konnektoren. Dieser Befehl kann nur in dem Fenster ausgeführt werden, in dem der Kanal erzeugt wurde. In Unternetzen kann dieser Kanal, der ja dort als Schnittstelle zum hierarchisch höher gelegenen Netz mit schwächer gezeichneter Ikone dargestellt wird, nicht gelöscht werden.

7.3.4 Transitions-Menü

naming comment

Es öffnet sich ein Text-Fenster, in dem ein benennender Kommentar für die angewählten Transition eingegeben werden kann. Über diesen kann die Transition im Netz identifizeirt werden.

purpose description
Hier kann, falls im Einzelfall sinnvoll, eine Beschreibung an die Transition angehängt werden, die ihren Sinn und Zweck angibt. Hierzu gehören z.B. eine Beschreibung, welche Funktionalität des realen Systems durch die Transition dargestellt wird, was die temporären Variablen bedeuten, welche Schnittstellen die Transitioncodes bearbeiten, was die Kopplung mit anderen Netzelementen bewirkt, usw.

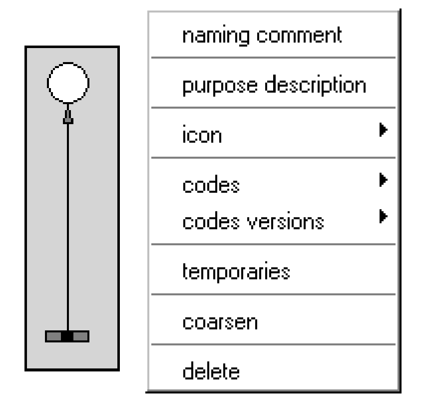


Abb. 7-7: Das Transitions-Menü

Die ‘purpose description’ kann dazu verwendet werden, um eine umgangssprachliche Beschreibung der Transition und ihrer Aufgabe im Netz niederzulegen und damit auch die Entwurfsgedanken im Modell zu verankern. Diese können bei späteren Erweiterungen sehr wertvoll sein, insbesondere dann, wenn das Modell längere Zeit nicht bearbeitet wurde und/oder neue Mitarbeiter eingearbeitet werden sollen.

icon

Erlaubt die Wahl, die Definition und die Änderung der Ikone für die Transition. Nach der Anwahl dieser Funktion erscheint ein Folgemenü, das weiter unten beschrieben wird.

codes
Bei der Auswahl dieser Funktion wird das Transition-Code-Folgemenü angezeigt, das im nächsten Abschnitt beschrieben wird.

temporaries
Es öffnet sich ein Fenster, in dem temporäre Smalltalk-Variablen für die angewählte Transition deklariert werden können. Temporäre Variablen sind lokale Variablen einer Transition und können innerhalb aller drei Code-Typen eingesetzt werden. Sie "überleben" die Zeitverzögerung der Transition. Dies bedeutet, daß die Werte, die ihnen durch den Bedingungs- oder Delay-Code zugewiesen wurden, beim Ausführen des Aktions-Codes immer noch gültig sind.

Die Alternative zu dieser Art, temporäre Variablen zu definieren, wird durch den interaktiven Code-Parser von Smalltalk angeboten. Dieser ermöglicht die interaktive Definition von temporären Variablen während der Übersetzung, sobald er eine nicht vereinbarte Variable findet.

Es sollte beachtet werden, daß temporäre (besser: lokale) Variablen explizit gelöscht werden müssen, wenn sie infolge von Code-Änderungen nicht mehr benötigt werden. Ansonsten wird unnötig Speicherplatz und Rechenzeit verbraucht, weil temporäre Variablen dynamisch erzeugt und vernichtet werden.

coarsen
Siehe dazu 'Menü für mehrere Elemente', Abschnitt 7.3.12.

delete
Löscht nach Bestätigung durch den Benutzer die Transition und alle damit verbundenen Konnektoren.

7.3.5 Transition-Codes-Menü

condition code

Die Funktion öffnet ein Text-Fenster, in dem der Bedingungs-Code für die angewählte Transition spezifiziert werden kann.

Der Bedingungs-Code ist eine Sequenz von Smalltalk-Anweisungen, die als Ergebnis einen der Wahrheitswerte 'true' oder 'false' liefern muss. Wenn kein Code angegeben ist, wird als Default-Wert 'true' erzeugt. Alle Konnektor-Variablen und temporären Variablen können im Bedingungscode verwendet werden.

Der Smalltalk-Text wird automatisch um die Deklarationen der Variablen erweitert und auf seine syntaktische Korrektheit hin überprüft. Tritt ein Fehler auf, so wird dieser entweder durch Einfügen von Fehlertext in den Smalltalk-Text markiert oder in einem Fehlerfenster mitgeteilt, das die weitere Vorgehensweise festlegt. Wird im Text-Fenster um Fehler-Kommentare erweiterter Smalltalk-Text angezeigt, so kann durch Bestätigen mit der linken Maustaste der ursprüngliche (fehlerhafte) Smalltalk-Text wieder angezeigt werden.

condition code
delay code
action code

Abb. 7-8: Das Transition-Code-Menü

delay code
Diese Funktion öffnet ein Textfenster für Smalltalk-Anweisungen, über die das Feuern der markierten Transition zeitlich verzögert werden kann. Der Delay-Code muß als Ergebnis eine nicht-negativen Zahl oder 'nil' liefern. Ist der Rückgabewert 'nil' oder wurde kein Delay-Code spezifiziert, so wird die Transition zur aktuellen Simulationszeit gefeuert (keine Verzögerung).

action code
Diese Funktion öffnet ein Fenster für den sog. Aktionscode, mit dem die durchzuführende Arbeit (Aktion) mit Smalltalk-Anweisungen festgelegt werden kann. Der Aktions-Code wird abgearbeitet und ausgewertet, sobald die Transition feuert, das heißt, sobald eine ggf. vorgegebene Verzögerung verstrichen ist. Der Aktions-Code wird

eingesetzt, um die Logik des Netzes zu erweitern, den Output-Variablen Werte zuzuweisen, um Attribute zu verändern, oder um gewisse Seiteneffekte zu bewirken (z.B. um eine Meldung auf den Bildschirm zu bringen, um eine Benutzerfunktion aufzurufen, um eine globale Variable zu ändern, usw.).

7.3.6 Versionshaltung für Transitionscodes

Gelegentlich ist es nützlich, wenn verschiedene Versionen eines Transitions-Codes verfügbar sind, sei es, daß man verschiedene Varianten einer Transition in einem PACE-Imagefile halten will oder daß man bestimmte Erweiterungen des Codes mit Testausdrucken für eventuelle spätere Verwendung aufheben will.

Für diese Zwecke wurde eine einfache Versionshaltung bereitgestellt, die über den Menüpunkt „code versions“ angesprochen werden kann. Nach dem Anwählen des Menüpunkts

code versions

erscheint das Menü:

condition code
delay code
action code

Abb. 7-9: Auswahl des Transitions-Codes

Darin kann der Transition-Code, dessen Versionshaltung bearbeitet werden soll, ausgewählt werden. Nach der Auswahl wird das zugeordnete Editfenster der Transition für den ausgewählten Code angezeigt.

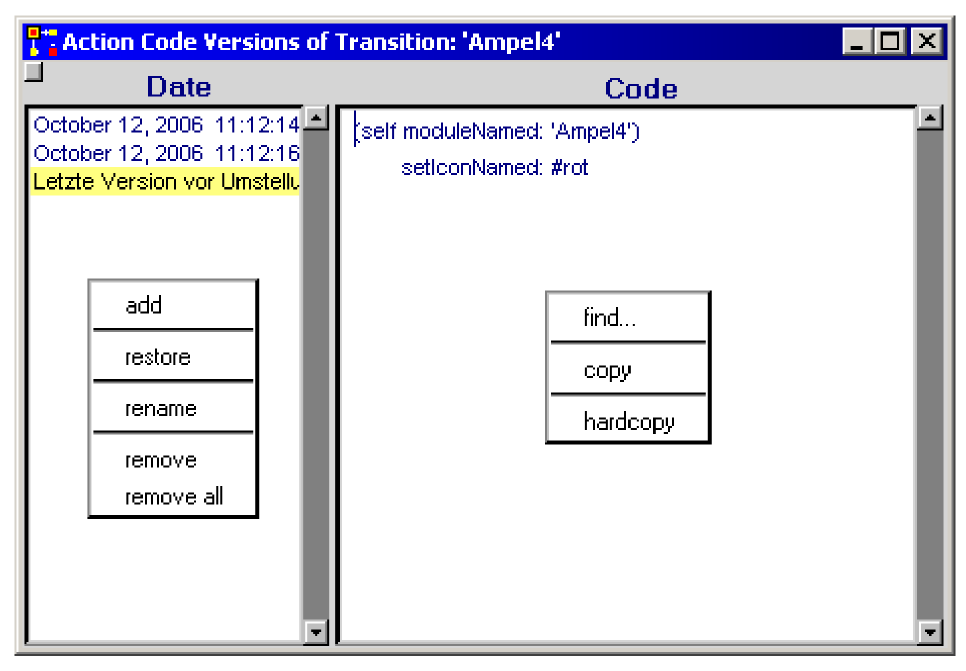


Abb. 7-10:Fenster für die Versionshaltung mit Menüs

Im linken Teilfenster wird defaultmäßig das Datum und die Uhrzeit des Eintrags notiert. Diese Zeitangabe kann vom Anwender durch eine beliebige andere Bezeichnung ersetzt werden. Wird eine Bezeichnung im linken Teilfenster markiert, so wird im rechten Teilfenster der zugehörige Transitionscode angezeigt.

<u>Menü im linken Teilfenster:</u>

add Die aktuelle Version des ausgewählten Transitionscodes wird als nächster Eintrag in die Versionshaltung aufgenommen. Es öffnet sich ein Abfragefenster für die Bezeichnung der Version. Wird nichts eingegeben, so wird in das linke Fenster das Datum und die Uhrzeit, zu dem dieser Eintrag erfolgt ist, eingetragen.

restore Der aktuelle Transitionscode wird mit dem zu dem markierten Eintrag gehörigen Transitionscode überschrieben.

rename Der Anwender kann die markierte Bezeichnung durch eine beliebige andere Bezeichnung ersetzen.

remove Die markierte Version wird in der Versionshaltung gelöscht.

remove all Alle gespeicherten Versionen werden in der Versionshaltung gelöscht.

Menü im rechten Teilfenster:

find Es öffnet sich ein Fenster, in dem der zu suchende Text eingegeben wird. Der gefundene Text wird markiert.

copy Der markierte Text wird im Zwischenspeicher gespeichert und kann an anderer Stelle mit dem paste-Kommando wieder eingesetzt werden.

hardcopy Der Text wir auf einen Drucker ausgegeben.

7.3.7 Modul-Menü

name
Um die Lesbarkeit einer PACE-Spezifikation zu erhöhen, kann mit dieser Funktion in einem Textfenster der Name des und/oder ein Kommentar über den Kanal festgelegt oder geändert werden.

In einem Textfenster kann der Name des Moduls festgelegt bzw. geändert werden. Um die Lesbarkeit einer PACE-Spezifikation zu erhöhen, sollten sinnvolle (dem jeweiligen Kontext angepaßte) Namen vergeben werden. Dabei sollte beachtet werden, daß der angegebene Name bei der Speicherung des Moduls oder bei der Codegenerierung für das Modul zur Bildung von Filenamen verwendet wird. Der Name sollte deshalb den Konventionen des jeweiligen Betriebssystems genügen und keine Leerzeichen enthalten. Ist dies nicht der Fall, so wird aus dem angegebenen automatisch ein der

Konvention des Betriebssystems entsprechender Name erzeugt, den der Benutzer bestätigen oder verwerfen muß.

purpose description
Hier kann eine Beschreibung an den Modul angehängt werden, aus der hervorgeht, welche Gründe zu seiner Einführung geführt haben und welche Aufgaben in den Modul integriert wurden. Die ‘purpose description’ ist besonders bei Moduln wichtig, um die Design-Gedanken bei der Strukturierung des Modells zu dokumentieren.

Die ‘purpose description’ kann dazu verwendet werden, um eine umgangssprachliche Beschreibung niederzulegen und damit auch die Entwurfsgedanken im Modell zu verankern. Diese können bei späteren Erweiterungen sehr wertvoll sein, insbesondere dann, wenn das Modell längere Zeit nicht bearbeitet wurde und/oder wenn neue Mitarbeiter eingearbeitet werden sollen.

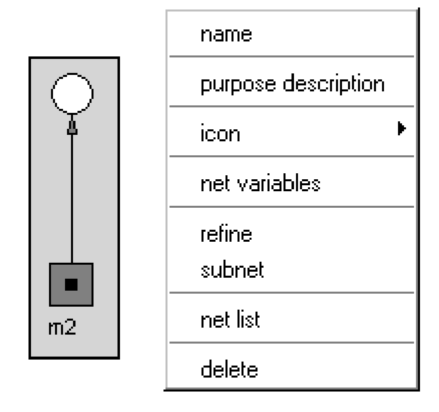


Abb. 7-11: Das Modul-Menü

icon
erlaubt die Wahl, die Definition und die Änderung der Ikone für das Modul. Nach der Anwahl dieser Funktion erscheint ein Folgemenü, das weiter unten beschrieben ist.

variables
Öffnet ein Fenster zur Definition der Modul-Variablen (siehe Abschnitt 3.7.3, ‘Modul-Variablen’).

refine
Dieses ist die inverse Operation zu 'coarsen'. Das angewählte Modul wird in seine einzelnen Elemente aufgelöst und in das hierarchisch höhere Netz integriert.

subnet
Öffnet ein Netz-Fenster, in dem das Modul-Netz dargestellt wird.

net list
Durch Ausführen dieses Befehles wird das Fenster mit der Netz-Liste in den Vordergrund gebracht. Der Cursor wird auf den Namen des Subnetzes gestellt, in dem das angewählte Modul vorkommt.

delete
Löscht mit Bestätigung durch den Benutzer das Modul mit allen darin enthaltenen Subnetzen und damit verbundenen Konnektoren.

7.3.8 Ikonen-Menü für S- und T-Elemente

Nachdem für ein S- oder für ein T-Element die 'icon'-Funktion aufgerufen wurde, erscheint das folgende Menü:

standard
scale
individual
alternate

Abb. 7-12:Das Ikonen-Menü

standard
An Stelle der angewählten Ikone wird die Standard-Ikone gesetzt. Diese kann für alle Netz-Elemente in der Netz-Liste über den

Menüpunkt ‘support’ der PACE-Hauptleiste, 'icon'-Untermenü und darin ‘default icon’ (siehe Kap. 5) angezeigt bzw. definiert werden.

scale
Mit dieser Funktion kann die Größe der angewählten Standard-Ikone verändert werden. Es erscheint eine Dialog-Box, in welcher der neue Darstellungsmaßstab in Bezug zur Voreinstellung angegeben werden kann.

Zwei verschiedene Eingabeformate sind vorgesehen:

- Die Angabe einer Zahl bewirkt eine Skalierung um diesen Faktor in der x- und y-Richtung.
- Falls die Skalierung in x- und y-Richtung unterschiedlich sein soll, werden die Skalierungen in einer Punkt-Notation angegeben: zahl1 @ zahl2 (Beispiel: 5@3.5). Zahl 1 bestimmt die Skalierung in horizontaler Richtung. Zahl 2 die Skalierung in vertikaler Richtung. Verzerrte Standard-Ikonen werden erreicht, wenn Zahl 1 ungleich Zahl 2 gewählt wird.

individual
An Stelle der angewählten Ikone wird eine individuelle, vom Benutzer selbst definierte Ikone gesetzt. Es erscheint ein Auswahlmenü mit den Namen der zur Verfügung stehenden Ikonen.

alternate
An Stelle der angewählten Ikone wird die Alternativ-Ikone gesetzt. Diese kann für alle Netz-Elemente in der Netz-Liste (über den entsprechenden Punkt im 'icon'-Menü, siehe oben) definiert werden.

7.3.9 Input-T-Konnektor-Menü

attributes
Öffnet ein Fenster zur Definition der Konnektor-Attribute (siehe Abschnitt 7.2.2)

copy attributes
Kopiert die Konnektor-Attribute in den Paste-Buffer.

paste attributes

Fügt den Code aus dem Inhalt des Paste-Buffers in die Attribut-Liste des angewählten Konnektors ein.

invert
Diese Funktion invertiert den Konnektor zu einem Inhibitor. Er funktioniert wie ein Schalter. Ein nochmaliges Betätigen ändert den zum Inhibitor invertierten Konnektor wieder in einen Input-T-Konnektor.

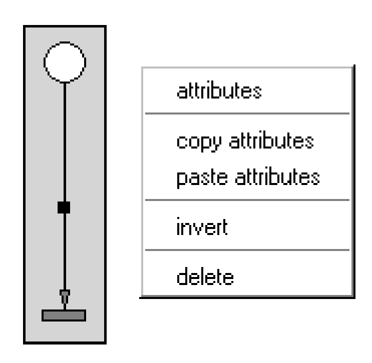


Abb. 7-13: Das Input-T-Konnektor-Menü

delete
Löscht nach Bestätigung durch den Benutzer den Konnektor.

7.3.10 Output-T-Konnektor-Menü

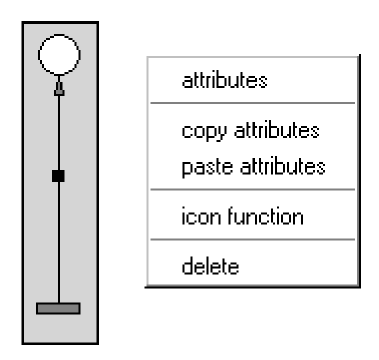


Abb. 7-14: Das Output-T-Konnektor-Menü

attributes
Die Funktion öffnet ein Fenster zur Definition der Konnektor-Attribute (siehe Abschnitt 7.2.2).

copy attributes
Kopiert die Konnektor-Attribute in den Paste-Buffer.

paste attributes
Fügt den Code aus dem Inhalt des Paste-Buffers in die Attribut-Liste des angewählten Konnektors ein.

icon function
Die Funktion öffnet ein Fenster, in dem die Ikonen-Funktion definiert werden kann. Eine Ikonen-Funktion ist ein Smalltalk-Block mit einem einzigen Argument.

Jedesmal wenn der Markenfluß über diesen Konnektor animiert werden soll, wird diese Funktion mit der Marke als Argument aufgerufen. Die Funktion definiert den Namen der Marken-Ikone, die dargestellt werden soll als Smalltalk-Symbol (Name mit vorangestelltem #-Zeichen). Wurde keine Funktion spezifiziert oder ist ihr Rückgabewert 'nil', so wird die Standard-Ikone dargestellt.

Wurde ein Ikonen-Name angegeben, so wird die Ikone in den Ikonen-Wörterbüchern des Modells oder der Teil-Netze gesucht, in denen der angewählte Konnektor vorkommt. Die Suche beginnt in dem Teil-Netz, in dem der Konnektor erzeugt wurde. Wird darin die gesuchte Ikone nicht gefunden, geht die Suche den ganzen Hierarchiebaum hinauf bis zu den Ikonen, die für das gesamte Modell gelten. Wird auch dort die gesuchte Ikone nicht gefunden, so kommt die Standard-Ikone zur Darstellung.

delete
Löscht den Konnektor, nachdem der Benutzer dies bestätigt hat.

7.3.11 Ikonen-Funktions-Menü

Im Ikonen-Funktions-Menü für Output-Konnektoren stehen folgende Funktionen zur Verfügung:

edit
Die Funktion öffnet ein Fenster, in dem die Ikonen-Funktion definiert werden kann. Diese muß ein Smalltalk-Block mit einem Argument sein. Der Wert des Blocks muß entweder ein Symbol oder 'nil' sein.

Eine Ikonen-Funktion kann gelöscht werden, indem ein leerer Text akzeptiert wird

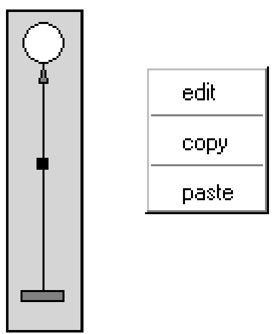


Abb. 7-15: Das Ikonen-Funktions-Menü

Beispiel:
In der Ikonen-Funktion von Abb. 7-16 hängt die darzustellende Ikone vom Wert des ersten Marken-Attributes ab. Ist der Attributwert größer als 10, wird die Ikone mit dem Namen '#large' dargestellt, sonst die mit dem Namen '#small'. Dabei sollte beachtet werden, daß in diesem Beispiel die Marke mindestens ein Attribut aufweisen muß, das die Botschaft ">" versteht. Ist dies nicht der Fall, so wird ein Laufzeitfehler gemeldet.

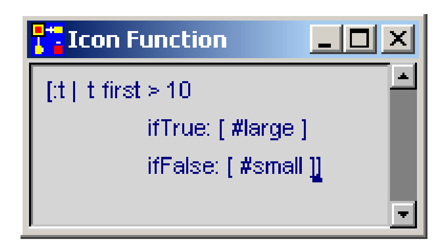


Abb. 7-16:Das Text-Fenster für eine Ikonen-Funktion

copy
Kopiert die Ikonen-Funktion in den Paste-Buffer.

paste
Weist die Ikonen-Funktion aus dem Inhalt des Paste-Buffers dem angewählten Output-Konnektor zu.

7.3.12 M-Konnektor-Menü

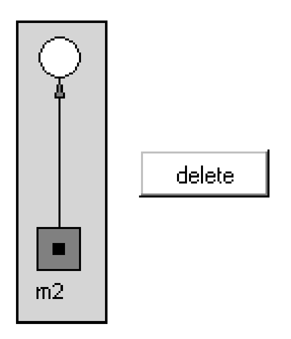


Abb. 7-17: Das M-Konnektor-Menü

Wird ein Konnektor angewählt, der von oder zu einem verfeinerten Element führt, und wird danach die rechte Maustaste gedrückt, so erscheint das in Abb. 7-17 dargestellte einzeilige Menü.

delete
Löscht nach Bestätigung durch den Benutzer den Konnektor.

7.3.13 Menü für mehrere Elemente (ohne Konnektoren)

Wurden mehrere Netz-Elemente, aber keine Konnektoren, markiert (siehe Kapitel 9, 'Arbeiten im Netzfenster') und wird danach die rechte Maustaste gedrückt, erscheint das in Abb. 7-18 dargestellte Menü.

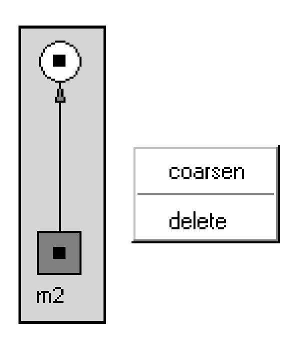


Abb. 7-18: Das Menü bei mehreren angewählten Elementen (ohne Konnektoren)

coarsen
Faßt die angewählten Netz-Elemente zusammen. Wurden ausschließlich S-Elemente selektiert, werden diese in einem Kanal zusammengefaßt. Wurden hingegen S- und T-Elemente angewählt, entsteht aus ihnen ein Modul, wobei die selektierten S-Elemente, welche die Schnittstelle bilden, nicht in das Modul aufgenommen werden.

Die zu 'coarsen' inverse Funktion heißt 'refine' und ist im Menü der verfeinerten Netz-Elemente (Modul und Kanal) zu finden.

delete
Löscht nach Bestätigung durch den Benutzer die angewählten Netz-Elemente.

7.3.14 Menü für mehrere Elemente (mit Konnektoren)

Wurden mehrere Netz-Elemente, inklusive Konnektoren, angewählt, und wird die rechte Maustaste gedrückt, erscheint folgendes Menü:

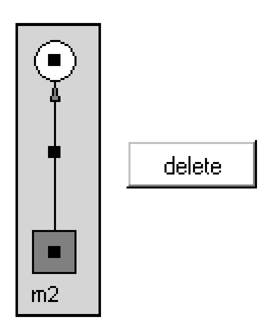


Abb. 7-19: Das Menü bei mehreren angewählten Elementen (mit Konnektoren)

delete
Löscht nach Bestätigung durch den Benutzer die angewählten Netz-Elemente.

7.3.15 Inskriptionen-Menü für S- und T-Elemente

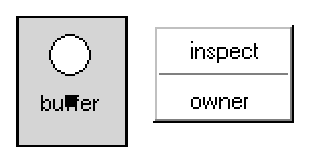


Abb. 7-20: Das Inskriptionen-Menü für S- und T-Elemente

Wird eine Inskription eines S- oder T-Elements selektiert und wird danach die rechte Maustaste gedrückt, so erscheint das in Abb. 7-20 gezeigte Menü.

inspect
Öffnet einen Smalltalk-Inspector, in dem die angewählte Inskription angezeigt wird und verändert werden kann. Die Inskription kann auch angezeigt werden, indem die entsprechende Funktion (z.B. 'naming comment' bei Transitionen) im Menü eines Netz-Elementes ausgeführt wird.

owner
Plaziert den Cursor auf das Element, zu dem die markierte Inskription gehört.

7.3.16 Inskriptionen-Menü für Konnektoren

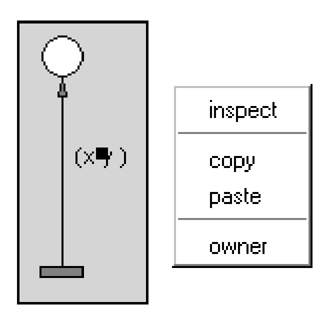


Abb. 7-21: Das Inskriptionen-Menü für Konnektoren

Selektieren einer Konnektor-Attributlist und Drücken der rechten Maustaste zeigt das in Abb. 7-21 dargestellte Menü.

inspect
Öffnet ein Fenster mit einer Auflistung aller Attribute des angewählten Konnektors.

copy
Kopiert die Konnektor-Attribute in den Paste-Buffer.

paste
Fügt den Code aus dem Inhalt des Paste-Buffers in die Attributs-Liste eines angewählten Konnektors ein.

owner
Plaziert den Cursor auf den Konnektor zu dem die markierte Inskription gehört.

7.4 Kanal-Unternetz-Menüs

Die Menüs, die in einem Kanal-Unternetz zur Verfügung stehen, unterscheiden sich zum Teil von denen, die in anderen Netz-Fenstern (für Hauptnetze oder Modul-Unternetze) zur Anwendung kommen. In der folgenden Beschreibung werden nur die Unterschiede behandelt. Die anderen Menüpunkte entsprechen in ihrem Verhalten den gleichnamigen Punkten aus anderen Menüs.

7.4.1 Hauptmenü im Kanal-Unternetz

Im Kanal-Unternetz können keine T-Elemente hinzugefügt werden. Wurde kein Element angewählt, erscheint durch Drücken der mittleren Maustaste das in Abb. 7-22 dargestellte Menü.

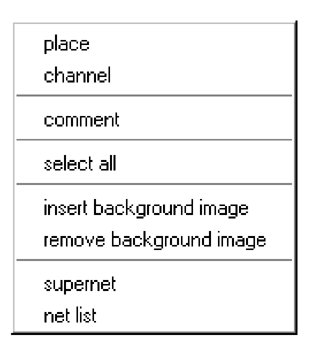


Abb. 7-22:Das Hauptmenü im Kanal-Unternetz

Die angezeigten Funktionen sind schon in früheren Menüs beschrieben worden.

7.4.2 Modul-Menü im Kanal-Unternetz

Die Funktionen **name** und **icon** wurden schon früher beschrieben.

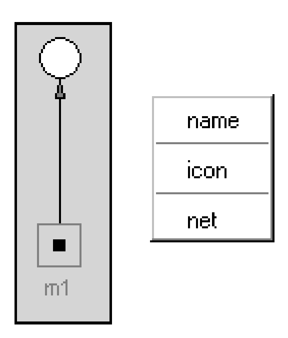


Abb. 7-23: Das Modul-Menü im Kanal-Unternetz

net
Dieser Befehl öffnet oder aktiviert ein Fenster mit dem Netz, in dem das angewählte Modul ursprünglich erstellt wurde. Der Cursor steht auf dem entsprechenden Modul.

7.5 Austauschen des S-Elementes eines Konnektors

Falls das S-Element (Stelle oder Kanal) eines Konnektors, dessen T-Element eine Transition oder ein Modul ist, gegen ein anderes S-Element ausgetauscht werden soll, so kann wie folgt vorgegangen werden:

Mit der linken Maustaste die Konnektorlinie anklicken und die Maustaste niedergedrückt halten. Mit der niedergedrückten, linken Maustaste den Cursor auf das neue S-Element plazieren. Bei diesem Vorgang wird an Stelle des Cursors ein Fadenkreuz angezeigt, sobald der Cursor den angewählten Konnektor verlassen hat. Befindet sich das Fadenkreuz auf dem neuen S-Element, kann die Maustaste wieder losgelassen werden. Das neu angewählte S-Element ist nun mit dem T-Element verbunden.

Man beachte, daß Verbindungen von und zu Modulen und Kanälen Zusammenfassungen von Konnektoren darstellen. Umhängen bewirkt, daß alle Einzelkonnektoren umgehängt werden. Die Attribute des Konnektors bleiben bei diesem Vorgang erhalten. Diese Vorgehensweise ist beispielsweise sehr nützlich, wenn das gleiche Unternetz innerhalb eines Moduls mehrmals gebraucht wird.

8 SIMULATOR

Im Simulations-Modus kann ein mit dem graphischen Editor erstelltes, syntaktisch korrektes Netz, auch wenn es noch unvollständig ist, animiert und dabei sein dynamisches Verhalten analysiert werden. Der Benutzer kann dabei jederzeit die Simulation anhalten, das Netz verändern, und danach entweder einen neuen Simulationslauf starten oder den unterbrochenen Simulationslauf fortsetzen.

Der Simulator wird durch Betätigen des 'simulator'-Schalters unten links in einem Netzfenster eingeschaltet.

8.1 Simulator-Menüs

Die Menüs, die in einem Simulations-Fenster angezeigt werden, sind von den jeweils angewählten Elementen abhängig. Wurde kein Element angewählt, erscheint durch Drücken der mittleren Maus-Taste das Hauptmenü des Simulators (das sog. No-Selection-Menü). Im Simulationsmodus kann, im Gegensatz zum Editiermodus, nur jeweils ein einziges Element zu einem Zeitpunkt angewählt werden. Mit dem Simulator-Hauptmenü kann die Simulation gestartet oder beeinflußt werden.

8.1.1 Simulator-Hauptmenü

initialze + run
Führt die Kommandos ‘initialize’ und ‘run’ hineinander aus.

Initialize + background run
Führt die Kommandos ‘initialize’ und ‘ background run’ hineinander aus.

run
Die Funktion startet, ausgehend vom momentanen Netz-Zustand, die Simulation des Ereignisflusses durch das geladene Netz. Dabei werden das Feuern der Transitionen und der Markenfluss durch Animation im Vordergrund sichtbar. Während der Simulation hat jede der drei Maustasten eine bestimmte Funktion (siehe weiter unten).

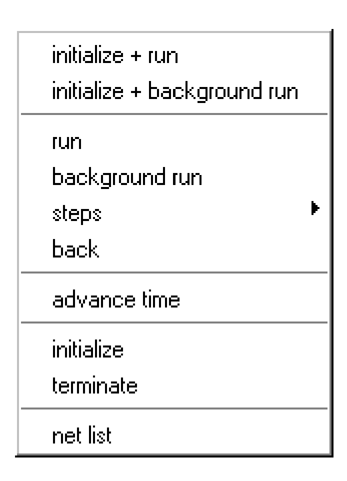


Abb. 8-1: Das Simulator-No-Selection-Menü

Die Simulation wird gestoppt, wenn entweder

- die linke Maustaste gedrückt wird oder wenn
- ein Breakpoint erreicht wird oder wenn
- keine Transition mehr feuerbar ist. In diesem Fall erscheint die System-Meldung "End of simulation“.

background run
Die Simulation der einzelnen Prozesse im spezifizierten Netz wird gestartet und mit größtmöglicher Geschwindigkeit ausgeführt. Das Feuern der Transitionen findet im Hintergrund statt, es wird nicht sichtbar dargestellt.

Da alle Inskriptionen ausgeführt werden, also auch solche, die Ikonen verändern, kann es auch in diesem Modus zu sichtbaren Veränderungen am Bildschirm kommen.

Die Simulation wird auf gleiche Weise wie im Animationsmodus angehalten (siehe oben).

step
Eine einzelne Transition soll gefeuert werden. Ist für diese Transition eine zeitliche Verzögerung ('delay') vorgegeben, so wird die Systemzeit um die angegebene Verzögerungszeit weitergestellt.

Es gibt vier Submenu Funktionen für Einzelschritte:

- step
  Der Einzelschritt wird ohne Fortsetzungs- und Stop-Code ausgeführt.
- step with continuation code
  Der Einzelschritt wird mit Fortsetzungs-Code ausgeführt.
- step with stop code
  Der Einzelschritt wird mit Stop-Code ausgeführt.
- step with continuation and stop code
  Der Einzelschritt wird mit Fortsetzungs- und Stop-Code ausgeführt.

back
Diese Funktion macht den zuletzt durchgeführten Schritt rückgängig. Durch Wiederholen von 'back' kann die Simulation bis zu einer wählbaren Tiefe rückwärts gefahren werden. Der Benutzer kann dies bestimmen, indem er den entsprechenden Wert für die Simulator-Option 'memory size' einträgt (siehe dazu Kapitel 5, Simulator-Optionen).

Es gibt viele Situationen, in denen das Rückverfolgen eines Simulationslaufs nützlich sein kann. Man muß sich aber darüber im Klaren sein, daß sich dabei nicht alle ausgeführten Operationen ungeschehen bzw. wieder rückgängig machen lassen. Wenn beispielsweise

eine Transitions-Inskription ein Objekt in eine Liste eingefügt hat, so wird das eingefügte Objekt durch einen Backtracking-Schritt nicht mehr aus der Liste entfernt. Sollten sich nach dem Ausführen einer 'back'-Funktion irgendwelche Probleme zeigen, so ist in der Regel ein neuer Simulationslauf unumgänglich. Besonders sei darauf hingewiesen, dass durch einen Rückwärtslauf das Rücksetzen von Statistiken auf den alten Stand nicht möglich ist und dass auch die Werte globaler Variablen nicht zurückgesetzt werden.

advance time
Mit dieser Funktion kann die Simulations-Zeit auf den nächsten Zeit-Wert in der Ereignis-Liste vorgestellt werden. Nach dem Durchführen der Funktion kann es vorkommen, daß Ereignisse zu einem Zeitpunkt, der vor der aktuellen Simulations-Zeit liegt, eingeplant sind. Diese Ereignisse werden beim Fortsetzen der Simulation zuerst bearbeitet.

initialize
Der im Editier-Modus definierte Initialzustand wird gesetzt und der 'initialization'-Code wird ausgeführt.

terminate
Dieser Befehl führt den 'terminate'-Code aus. Danach werden alle Marken aus dem Netz entfernt.

supernet
Dieser Menuepunkt wird nur angezeigt, wenn es ein in der Hierarchie höher liegendes Netz gibt.

Die Funktion öffnet oder aktiviert das im Hierarchiebaum höher liegende Netz-Fenster. Der Cursor wird auf die Ikone des Subnetzes eingestellt, von dem aus diese Funktion aufgerufen wurde.

net list
Durch Ausführen dieser Funktion wird das Fenster mit der Netz-Liste in den Vordergrund gebracht. Darin wird der Name des Netzes markiert, von dem aus diese Funktion aufgerufen wurde. Der Cursor wird im Netzfenster gezeigt.

8.1.2 Stellen-Menü

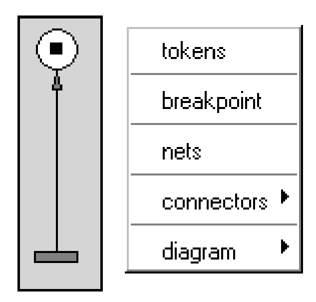


Abb. 8-2: Das Stellen-Menü

tokens
Dieser Befehl ermöglicht den Zugang zu allen Marken, die momentan auf der angewählten Stelle liegen, und zu all ihren Attributen. Es können interaktiv Marken hinzugefügt, verändert und gelöscht werden. Die Änderungen haben aber nur für den aktuellen Simulationslauf Gültigkeit.

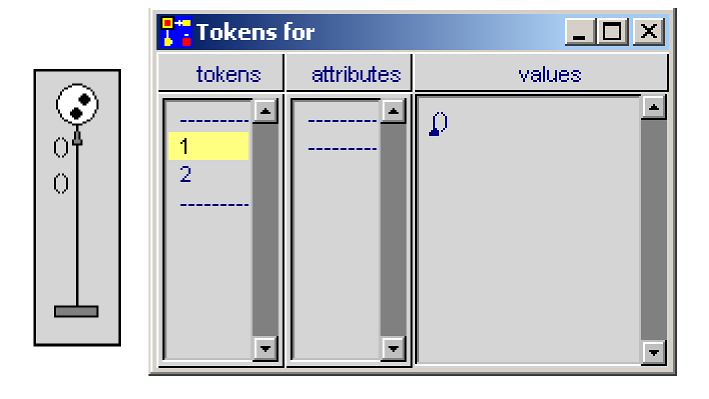


Abb. 8-3: Das Markenfenster im Simulator

breakpoint
Definiert und setzt eine Unterbrechung auf die angewählte Stelle. Diese wird der 'breakpoint'-Liste hinzugefügt. Ist einer Stelle eine Unterbrechung zugeordnet worden, so unterbricht dieser die Simulation noch bevor der Stelle eine Marke hinzugefügt oder von ihr abgezogen werden kann. Die für diese Stelle spezifizierte Unterbrechung wird bei einer Unterbrechung im 'breakpoints'-Fenster markiert. Weitere Erläuterung hierzu sind im Abschnitt 8.7: 'Unterbrechungspunkte' zu finden.

nets
Zeigt eine Liste aller Unternetze an, in denen die angewählte Stelle vorkommt. Durch Auswählen eines dieser Netze wird ein Fenster mit dem entsprechenden Netz geöffnet oder aktiviert. Sobald das Fenster geöffnet ist, springt der Cursor automatisch auf die angewählte Stelle.



Abb. 8-4: Netzauswahlfenster

connectors
Nach der Wahl dieses Menüpunktes erscheint ein Untermenü mit der Auflistung aller möglichen Konnektorarten. Sobald eine Auswahl getroffen wurde, erscheint eine Liste aller Konnektoren des entsprechenden Typs zu dieser Stelle. Das Anwählen eines Listenpunktes

bewirkt, daß ein Fenster mit dem Netz, welches diesen Konnektor enthält, geöffnet oder aktiviert wird. Sobald das Netz-Fenster geöffnet ist, springt der Cursor automatisch auf die entsprechende Stelle.



Abb. 8-5: Konnektorauswahl

diagram
Öffnet ein Diagramm-Fenster für die angewählte Stelle. Darin können statistische Auswertungen beauftragt werden. Bei jeder Zustandsänderung der Stelle werden Statistikdaten gemäß der programmierten Funktion gesammelt und angezeigt. Siehe dazu die Ausführungen im Abschnitt 'Zeitabhängige Diagramme' weiter unten.

8.1.3 Kanal-Menü

subnet
Öffnet oder aktiviert ein Fenster, in dem die Elemente des angewählten Kanals dargestellt werden.

nets
Zeigt eine Liste aller Teilnetze an, in denen der selektierte Kanal vorkommt. Durch Auswählen eines dieser Netze wird ein Fenster mit dem entsprechenden Netz geöffnet oder aktiviert. Sobald das Fenster geöffnet ist, springt der Cursor automatisch auf den entsprechenden Kanal.

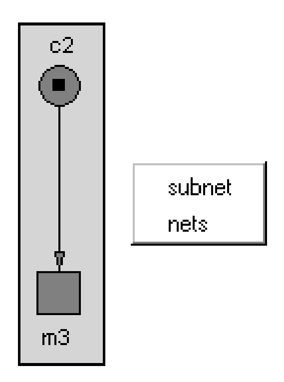


Abb. 8-6: Das Kanal-Menü

8.1.4 Transitions-Menü

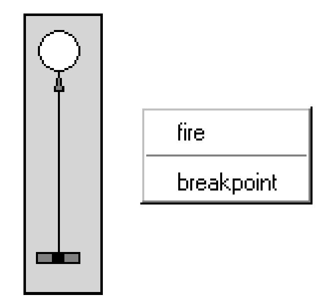


Abb. 8-7: Das Transitions-Menü

fire
Die angewählte Transition soll feuern, sofern dies gemäß Spezifikation und Systemdefinition überhaupt möglich ist, d.h. sofern sie aktiviert wurde. Weist die Transition einen 'delay' auf, wird sie diesem Wert entsprechend in die Ereignis-Liste eingetragen und bei Erreichen des Zeitpunkts gefeuert.

breakpoint
Definiert und setzt eine Unterbrechung auf die angewählte Transition. Diese wird der 'breakpoint'-Liste hinzugefügt. Befindet

sich ein Breakpoint auf einer Transition, wird die Simulation vor dem Feuern der Transition angehalten. Bei Auftreten einer Unterbrechung wird die für diese Stelle spezifizierte Unterbrechung im 'breakpoints'-Fenster markiert. Weitere Ausführungen zu diesem Thema sind weiter unten im Abschnitt 'Unterbrechungspunkte' zu finden.

8.1.5 Modul-Menü

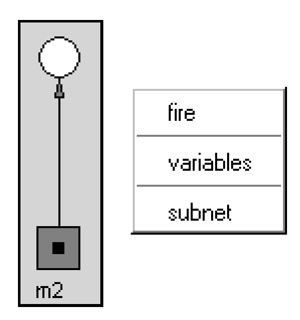


Abb. 8-8: Das Modul-Menü

fire
Feuert eine Transition im Unterbaum des angewählten Moduls.

variables
Zeigt ein Fenster mit den Modul-Variablen an.

subnet
Öffnet oder aktiviert ein Fenster, in dem das Modul (Unternetz) angezeigt wird.

8.1.6 T-Konnektor-Menü

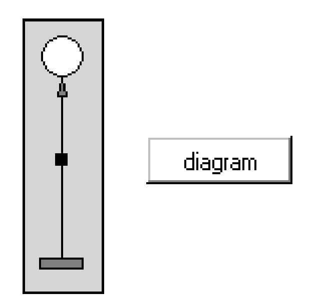


Abb. 8-9: Das T-Konnektor-Menü

diagram
Öffnet ein Diagramm-Fenster für den angewählten Konnektor (siehe auch Abschnitt 'Zeitabhängige Diagramme' weiter unten).

8.2 Aktivieren von gebundenen Fenstern (fixed windows)

Gebundene Fenster sind, wie es der Name schon sagt, immer an das gleiche Teilnetz gebunden. Oft sind Fenster für mehrere Teilnetze gleichzeitig geöffnet. Der Simulator springt dann von einem Netzfenster zum anderen. Dabei aktiviert er jeweils das Fenster (und bringt es an in den Vordergrund der Windows-Oberfläche), in dem gerade eine Transition feuert. Der Simulationsablauf wird nur in den Fenstern permanent dargestellt, die gebunden (fixed) sind und in denen der Simulations-Modus eingeschaltet ist.

Im aktiven Fenster wird der Markenfluss der feuernden Transition animiert. Das aktive Fenster muss dabei nicht unbedingt die feuernde Transition und alle mit ihr verbundenen Stellen beinhalten. Diese können in einem anderen Teilnetz (Modul) vereinbart sein. Es kann sogar sein, dass gar keine ihrer Input- oder Output-Stellen darin enthalten ist.

8.3 Animation in nicht gebundenen Fenstern (non fixed windows)

Beim Beginn einer Simulation wird in einem nicht gebundenen Netz-Fenster immer das Teilnetz angezeigt, in dem sich die feuernde Transition befindet. Sofern ein Teilnetz nicht in einem weiteren gebundenen Fenster dargestellt ist, tauscht der PACE-Simulator während des Simulationsablaufs automatisch das dargestellte Teilnetz gegen das Teilnetz aus, in dem sich die jeweils feuernde Transition befindet. Damit kann das dynamische Verhalten eines Modelles ganzheitlich angezeigt werden; man verpasst deshalb kein Feuern.

Bei größeren Systemen wird man bei dieser Vorgehensweise bald den Überblick verlieren. Um die Übersicht zu behalten und die Animation auf die gerade interessanten Teile eines Modells zu beschränken, kann man wie folgt vorgehen:

Man schränkt die Zahl der Module, die betrachtet werden sollen, durch Ausführen der Funktion '...' (Drei-Punkte-Funktion) in der Netz-Liste ein. Dazu wird zunächst der Druckknopf 'enable ... in net list for animation' in den Simulator-Optionen bedient und die '...'-Option auf 'on' gesetzt. Der Simulator zeigt danach während der Simulation nur noch die Teil-Netze im Netz-Fenster an, deren Namen in der Netz-Liste angezeigt werden.

8.4 Funktionen der einzelnen Maustasten während der Simulation

Während des Ablaufens der Simulation haben die drei Maustasten andere Funktionen als beim sonstigen Arbeiten mit PACE.

8.4.1 Linke Maustaste

Durch Drücken der linken Maustaste wird die Simulation unterbrochen. Das Feuern der aktuellen Transition wird zunächst noch beendet. Deshalb muss die Maustaste so lange niedergedrückt gehalten werden, bis das letzte Feuern ganz abgeschlossen ist und der Cursor wieder angezeigt wird.

8.4.2 Mittlere Maustaste

Nach dem Drücken der mittleren Maustaste wird die Animation solange angehalten, wie die Taste niedergedrückt wird. Diese Option ist sehr nützlich, um mehr Zeit für das Studium der Attribute gefeuerter Marken zu haben und um Screen-Shots von den Netzfenstern zu machen.

8.4.3 Rechte Maustaste

Das Betätigen der rechten Maustaste unterdrückt die animierte, graphische Darstellung des Markenflusses und beschleunigt damit den Simulationslauf. Es wird sofort der neue Zustand nach dem Feuern einer Transition angezeigt. Bei Loslassen der rechten Maustaste wird der Markenfluss wieder wie oben erklärt dargestellt.

Diese Operation ist bei verschiedenen Betriebssystemen (z.B. Windows 95 und Windows 98) problematisch! Wird sie zu lange durchgeführt, so kann der sog. GDI-Speicher (GDI = Graphics Device Interface), der noch mit 16 Bits adressiert wird, überlaufen und das Betriebssystem beendet das Programm. Bei anderen echten 32-Bit-Betriebssystemen (z.B. Windows-2000, Windows-Xp,

Windows-Vista, Unix) ist diese Operation unproblematisch. Doch empfiehlt es sich auch hier, die Funktion nicht längere Zeit ununterbrochen ablaufen zu lassen.

8.5 Ausführungs-Status

Der Ausführungs-Status besteht aus den Daten, die vom Simulator verwaltet werden und aus der jeweiligen Stellenbelegung. Diese Daten können durch Ausführen der 'reset simulator'-Funktion (im 'simulator'-Menü der PACE-Hauptleiste) bereinigt werden. Das Ausführen der 'reset'-Funktion ist bei größeren Netzen vor dem Abspeichern des aktuellen PACE-Image nützlich, um nicht mehr benötigten Speicherplatz freizugeben.

Der Ausführungs-Status kann auf vier Arten initialisiert werden.

- Durch Ausführen des 'initialize'-Befehls (im Simulator-Hauptmenü).
- Durch Starten der Simulation eines Modells, dessen Anfangs-Stellenbelegung zuvor zurückgesetzt worden ist ('reset'-Funktion).
- Durch Starten der Erstsimulation eines Modells, das zuvor noch nie initialisiert worden ist.
- Durch Drücken der Start-Taste in einer PACE-Executive, die durch Funktionen im Simulator-Menü der PACE-Hauptleiste geöffnet werden.

8.6 Simulations-Algorithmus

Auch wenn ein Anwender von PACE nicht alle simulator-internen Details kennen muss, die bei einer Simulation auftreten, so sind doch einige Aspekte für das Verständnis der Verhaltensweise der Modelle während der Simulation nützlich. Der Simulations-Algorithmus, der dieses Verhalten bestimmt, wird im folgenden kurz erläutert.

8.6.1 Ereignis-Liste

Ein Ereignis ist definiert durch jeweils eine Transition, deren Input-Marken und den dazugehörigen Verbindungen. Die Reihenfolge, in der mögliche Ereignisse eintreten, wird durch die Departure-Einstellungen der Eingabestellen bestimmt. Der Simulator speichert alle Ereignisse, die entweder bereits vorüber oder für einen späteren Zeitpunkt eingeplant sind. Die Ereignis-Liste ist eine zeitlich sortierte Auflistung aller gespeicherten Ereignisse.

Jeder Eintrag in die Ereignis-Liste besteht aus der Zeit, zu der das betreffende Ereignis stattfinden soll und einem Satz von Transitionen, die zu diesem Zeitpunkt gefeuert werden sollen. Die einzelnen Elemente werden nach der Zeit, in aufsteigender Reihenfolge sortiert. Soll das Feuern einer Transition für einen bestimmten Zeitpunkt eingeplant werden, so durchsucht PACE die Ereignis-Liste nach einem Eintrag für diesen Zeitpunkt. Bei erfolgreicher Suche wird das Ereignis dem Satz von Ereignissen dieses Eintrags hinzugefügt. Verläuft die Suche ergebnislos, fügt PACE einen neuen Eintrag an der entsprechenden Stelle der Ereignis-Liste ein.

8.6.2 Simulations-Protokoll

Der Simulator zeichnet die letzten gefeuerten Transitionen in einer FIFO-Liste (FIFO - First In First Out) auf, die später beim Rückwärtsfahren des Simulators ausgewertet wird.

Der Benutzer kann bestimmen, wieviele Einträge diese Liste maximal aufweisen soll, d.h. wieviele Rückschritte maximal möglich sein sollen. Dazu ist wie folgt zu verfahren:

1. Ausführen der Funktion ‘Simulator’ in der PACE-Hauptleiste.

2. Ausführen der Funktion ‘options’.

3. Verändern des Werts von 'memory size' auf die gewünschte Listenlänge.

Bei der Festlegung der Listenlänge sollte beachtet werden, daß bei einer langen Liste mehr Informationen verarbeitet werden müssen, der Simulator also langsamer läuft. Eine kurze Liste führt zu einer schnell ablaufenden Simulation. Der Wert von 'memory size' ist mit 20 vorbesetzt, d.h. 20 Rückschritt sind möglich.

8.6.3 Simulations-Zeit

Durch Ausführen des 'initialize'-Befehls wird die Simulationszeit auf 0 (Null) gesetzt.

Alle im Netz zu einem Zeitpunkt feuerbereiten Transitionen werden zu diesem Zeitpunkt gefeuert. Ist im Netz keine feuerbare Transition mehr vorhanden, so wird die Simulationszeit auf den Wert des ersten Eintrages in der Ereignis-Liste gesetzt.

Der Simulator löscht die Transitionen, deren Feuerungen zu diesem Zeitpunkt eingeplant waren, aus der Ereignis-Liste und versucht, die Transitionen mit der entsprechenden Marken-Kombination zu feuern.

Der Simulationslauf ist beendet, sobald keine Transition mehr gefeuert werden kann, d.h. sobald die Ereignis-Liste leer ist.

8.6.4 Auflösung von Konflikten

Nach der Theorie können Konflikte entstehen, wenn eine Marke von mehr als einer Transition verbraucht werden kann. Praktisch werden solche Konflikte durch die deterministische Vorgehensweise von PACE aufgelöst, weil Konnektoren, die eine Marke von einer Stelle abziehen können, immer in der gleichen implementierungs-abhängigen Reihenfolge abgearbeitet werden. Der zuerst gefundene Konnektor, der die Marke weiterleiten kann, wird von PACE verwendet.

Es ist aber möglich, die in PACE vorbesetzte deterministische Vorgehensweise zu ändern. Die jeweils gewünschte Strategie kann mit der folgenden, in einem Workspace auszuführenden Meldung geändert werden:

UserPreferences randomScheduling: aBoolean

Dabei kann aBoolean entweder true (zufällig) oder false (deterministisch) sein.

Wird für eine bestimmte Stelle eine abweichende Abarbeitungs-Reihenfolge gewünscht, so muß diese vom Benutzer selbst modelliert werden. Soll beispielsweise bei deterministischer Vorgehensweise die Auswahl des weiterleitenden Konnektors zufällig sein, so könnte die Selektion über ein Marken-Attribut bewerkstelligt werden, dem ein Zufallswert zugewiesen wird. Die zulässigen Zufallswerte werden jeweils den Attributen der weiterleitenden Konnektoren zugewiesen.
Im PACE-Turorial ist ein Beispiel für die Änderung der Abarbeitungs-Reihenfolge unter Verwendung des Standard-Moduls Priority.sub aus der Modulbibliothek im Verzeichnis ‘modules’ beschrieben.

8.7 Unterbrechungspunkte

In PACE können zwei verschiedene Unterbrechungspunkte modelliert werden: Unterbrechungspunkte für Netz-Elemente (node breakpoints) und zeitlich bedingte Unterbrechungspunkte (time breakpoints). Für beide Arten legt PACE eine Liste aller spezifizierten Unterbrechungspunkte an.

Diese Listen können jeweils in einem Fenster dargestellt werden, das über das ‘debugger'-Menü der PACE-Hauptleiste geöffnet oder aktiviert werden kann. Darin können Unterbrechungspunkte aktiviert, deaktiviert und gelöscht werden.

Wird während der Simulation ein aktiver Unterbrechungspunkt erreicht, so wird das entsprechende Fenster geöffnet oder aktiviert. Sowohl die im Vordergrund animierte als auch die im Hintergrund ablaufende Animation wird unterbrochen.

Gelangt die Simulation zu einem deaktivierten Unterbrechungs-punkt, so hat dies keine Auswirkung auf die Simulation.

Die Listen der Unterbrechungspunkte werden zusammen mit einem Modell gespeichert.

8.7.1 Unterbrechungspunkte für Netz-Elemente (node breakpoints)

Bei der Beschreibung des Simulator-Menüs wird erläutert, wie Unterbrechungspunkte auf Stellen und Transitionen gesetzt werden können.

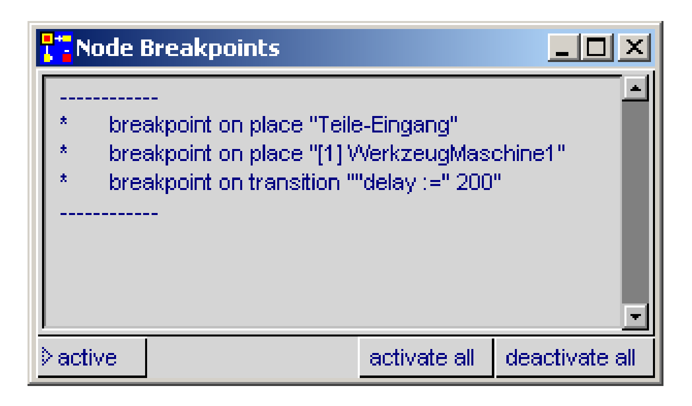


Abb. 8-10: Eine Unterbrechungsliste für Netzknoten

Abb. 8-10 zeigt ein Fenster für Unterbrechungen, die Netz-Knoten (Stellen, Transitionen) zugeordnet worden sind. Sobald eine Marke einen Netz-Knoten mit Unterbrechung passiert, wird der Ablauf des Simulators unterbrochen und die Kontrolle an den Benutzer übergeben. Das Fenster für Knoten-Unterbechungen kann während des Testens permanent angezeigt werden und wird automatisch auf den jüngsten Stand gebracht, sobald zusätzliche Unterbrechungs-Knoten vereinbart werden oder wenn Unterbrechungspunkte wegfallen.

Ist, wie in Abb. 8-10, keine Zeile im Knotenfenster selektiert, so wird bei Drücken der rechten Maus-Taste das folgende Menü angezeigt:



Abb. 8-11: Menü im Fenster 'Node Breakpoints', wenn keine Unterbrechung (Zeile) angewählt worden ist.

Wird die Funktion ausgeführt, so werden alle Unterbrechungen an Netzknoten gelöscht.

Wird eine Zeile im Knoten-Fenster selektiert, so erhält man durch Drücken der rechten Maus-Taste das folgende Auswahlmenü:

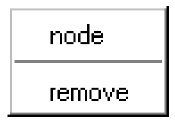


Abb. 8-12: Menü im Fenster 'Node Breakpoints', wenn eine Unterbrechung (Zeile) angewählt worden ist.

node
Öffnet oder aktiviert ein Fenster, in dem sich der Netz-Knoten mit der angezeigten Unterbrechung befindet. Der Cursor wird auf den Netz-Knoten platziert.

remove
Löscht die Unterbrechung.

Netzelemente mit Unterbrechungspunkten werden im Netzfenster besonders gekennzeichnet, indem sie nicht gefüllt werden, d.h. das Innere des Standard-Ikons stimmt mit dem Fenster-Hintergrund überein. Der Benutzer kann selbst festlegen, welche spezielle Farben er zur Darstellung von Netz-Elementen mit Unterbrechungen verwenden will. Diese werden, wie auch die Farben für andere Netz-Bestandteile, wie folgt definiert:

1. In der PACE-Hauptleiste das ‘view’-Menü aufrufen.
2. Darin die Funktion ‘colors’ ausführen.
3. Darin die Funktion ‘net constituents colors’ ausführen.
4. 'breakpoint node border' und/oder 'breakpoint node fill' neu besetzen.

8.7.2 Zeitlich bedingte Unterbrechungspunkte (time breakpoints)

Eine zeitliche Unterbrechung bewirkt, daß die Simulation unterbrochen wird, sobald der spezifizierte Simulations-Zeitpunkt erreicht ist.

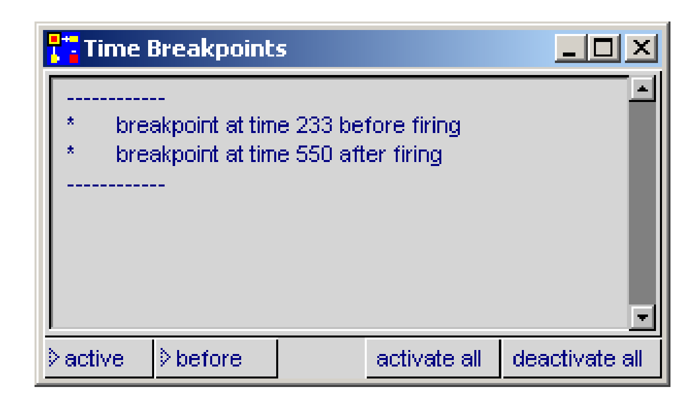


Abb. 8-13:Liste von Zeit-Unterbrechungen

Für jede zeitliche Unterbrechung können zwei Modi gewählt werden:

1. Unterbrechung, bevor überhaupt eine Transition zu diesem Zeitpunkt gefeuert wurde, oder

2. Unterbrechung, nachdem alle Transitionen, die für diesen Zeitpunkt vorgesehen sind, gefeuert haben.

Der jeweils gewünschte Modus wird gewählt, indem zunächst eine Zeile in der Unterbrechungsliste angewählt und dann der Druckknopf ‘before’ am unteren Rand des Fensters bedient wird.

Ist keine Zeile angewählt und wird die rechte Maus-Taste gedrückt, so erscheint folgendes Auswahlmenü:



Abb. 8-14:Menü im Fenster 'Time Breakpoints', wenn keine Unterbrechung (Zeile) angewählt worden ist.

add
Fügt der Liste eine weitere zeitliche Unterbrechung hinzu. Bei der Ausführung der Funktion geht ein Dialogfenster auf, in das die gewünschte Unterbrechungszeit einzutragen ist:



Abb. 8-15: Dialogfenster für die Eingabe der Unterbrechungszeit.

remove all
Löscht alle zeitlichen Unterbrechungen.

Ist im Fenster 'time breakpoints' eine Zeile selektiert, so wird beim Drücken der mittleren Maus-Taste das Menü:



Abb. 8-16: Menü im Fenster 'Time Breakpoints', wenn eine Unterbrechung (Zeile) angewählt worden ist.

angezeigt. Es besteht nur die Möglichkeit, die angewählte zeitliche Unterbrechung zu löschen.

8.8 Zeitabhängige Diagramme

PACE stellt im Simulator Funktionen zur Verfügung, die während der Simulation anfallende Daten in Form von Balken- und Liniendiagrammen darstellen. Hierzu wird für jedes anzuzeigende Diagramm ein eigenes spezielles Diagramm-Fenster geöffnet.

Diagramme können Stellen und T-Konnektoren zugeordnet werden. Der Benutzer kann die Diagramm-Fenster mit Hilfe der nachfolgend beschrieben Menüs bearbeiten.

8.8.1 Diagramm-Menü

Bei eingeschalteter Taste 'simulate' wird im Netz-Fenster eine Stelle oder ein T-Konnektor durch Anklicken mit der linken Maus-Taste ausgewählt und danach mit der mittleren Maus-Taste das zugeordnete Auswahlmenü angezeigt. Nach dem Ausführen der Funktion 'diagram' erscheint auf dem Bildschirm das in Abb. 8-17 dargestellte Menü. Die Funktionen des Untermenüs wurden in einem früheren Abschnitt schon erläutert.

Jeder der Menüpunkte öffnet ein spezielles typabhängiges Diagramm-Fenster. Dieses wird jedesmal, wenn die dem Diagramm zugeordnete Stelle bzw. der zugeordnete Konnektor während der Simulation ins Spiel kommt, auf den neuesten Stand gebracht. Nach welchen Kriterien dies geschehen soll, kann der Benutzer über die Funktion des Diagramms selbst bestimmen.

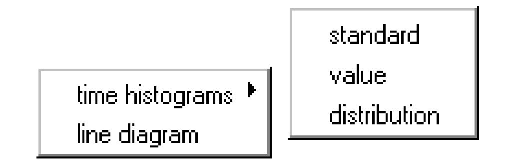


Abb. 8-17: Das Menü zur Auswahl des Diagrammtyp

8.8.2 Funktionen in den Diagramm-Fenstern

Die Funktion, welche die anzuzeigenden Daten errechnet, wird jedesmal aufgerufen, bevor sich der Zustand des zugeordneten Netz-Elementes ändert. Das Element selbst und die Zeitspanne, die seit seinem letzten Aufruf vergangen ist, liefern die auszuwertenden Input-Parameter für die Berechnung der Diagramm-Werte.

Jedem Element ist eine Default-Funktion zugeordnet, die einen Smalltalk-Code-Vorschlag in Form eines Smalltalk-Blocks enthält. Dieser Smalltalk-Block kann vom Benutzer beliebig geändert werden. Die dabei zu verwendenden Botschaften, die den Zugang zu den Marken einer Stelle ermöglichen, sind im Kapitel 3, 'Die PACE-Sprache' beschrieben.

Rückgabewert

Der Rückgabewert der Diagramm-Funktion ist ein Punkt. In Smalltalk wird ein Punkt generiert, indem die Botschaft '@' zur X-Koordinate mit der Y-Koordinate als Argument gesandt wird (z.B. 2 @ 3).

In Balkendiagrammen wird der Y-Wert eines Balkens je nach gewählter Funktion aus den Y-Koordinaten aller Punkte berechnet. Im values-Diagramm ist das die Summe der Y-Koordinaten aller Schnittpunkte, deren X-Koordinate sich innerhalb des X-Intervalls befindet. Eine Y-Koordinate kann dabei auch einen negativen Wert aufweisen.[11]

In Liniendiagrammen werden die Punkte in der Reihenfolge, in der sie durch die Funktion berechnet wurden, durch gerade Linien miteinander verbunden.

---

[11] Die beschriebenen Balkendiagramme lassen sich teilweise etwas umständlicher auch unter Verwendung der 'Allgemeinen Histogramme' aus Kapitel 6 realisieren

In den folgenden Abschnitten werden einige Diagramm-Funktionen für values-Diagramme beschrieben. Für die anderen Diagrammtypen gelten entsprechende Funktionen.

8.8.2.1 Default-Diagramm-Funktion für Balkendiagramme der Stellen

Vor der Zustandsänderung des zugeordneten Netz-Elementes wird berechnet, wie lange sich eine bestimmte Anzahl von Marken auf der Stelle aufgehalten hat.

```
[:marking :interval|
marking  tokenNumber@interval].
```

Dieses Diagramm zeigt für jedes X-Intervall, wie lange (Y-Wert) die Stelle mit der durch das X-Intervall angegebenen Anzahl von Marken belegt war.

8.8.2.2 Default-Diagramm-Funktion für Balkendiagramme der Konnektoren

```
[:aToken :interval | (aToken at: 1) @ 1]
```

Dieses Diagramm zeigt für jedes X-Intervall, wie oft (Y-Wert) der Wert des ersten Attributes der Marke innerhalb dieses Intervalls vorgekommen ist.

8.8.2.3 Default-Diagramm-Funktion für Liniendiagramme der Stellen

Bei Liniendiagrammen kann die Anzahl der Marken, die sich zum aktuellen Zeitpunkt auf Stelle befinden, erst bei dem nächsten Funktions-Aufruf angezeigt werden, weil sich die X-Koordinate erst durch Subtraktion der vergangenen Zeitspanne von der aktuellen Zeit ergibt.

```
[:marking :interval |
CurrentTime - interval @ marking tokenNumber]
```

Dieses Diagramm zeigt für jeden Zeitpunkt (X-Wert) die gesamte Anzahl der Marken, die sich auf der Stelle (Y-Wert) befinden.

8.8.2.4 Default-Diagramm-Funktion für Liniendiagramm der Konnektoren

```
[:aToken :interval |
CurrentTime @ (aToken at: 1)]
```

Dieses Diagramm zeigt den Wert des ersten Attributes der Marke (Y-Wert), der zum Zeitpunkt (X-Wert), an dem die Marke den Konnektor durchfließt, vorliegt.

8.8.2.5 Unwirksame Funktionsaufrufe

Erhält eine Diagrammfunktion den Rückgabewert 'nil', so hat der Aufruf der Diagramm-Funktion keine Auswirkung auf die Darstellung im Diagramm.

8.8.3 Beispiele von Funktionen für die diagrammatische Darstellung anderer Objekte

Außer den angegebenen, lassen sich beliebige andere Objekte als Funktion der Zeit darstellen. Im folgenden werden einige Beispiele angegeben.

8.8.3.1 Ausgabe von Marken-Attributen in ein Liniendiagramm

```
[:marking :interval |
CurrentTime - interval @ (marking token: i attribute: k) ]
```

Das k-te Attribute der i-ten Marke wird in das Liniendiagramm ausgegeben. Gibt es keine i-te Marke oder hat diese kein k-tes Attribut, so wird nichts ausgegeben.

8.8.3.2 Ausgabe von Attributen in ein Histogramm

[:marking :interval |
(marking token: i attribute: k) @ interval]

Das k-te Attribut des i-ten Token wird in ein Histogramm ausgegeben.

8.8.3.3 Ausgabe von Werten in ein Liniendiagramm

[:marking :intervall |
CurrentTime - interval @ variablenname]

Falls das Ergebnis von Ausdrücken ausgegeben werden soll, kann dieses entweder einer globalen Variable zugewiesen werden, die dann ausgegeben wird, oder ‘variablenname’ ist durch den Ausdruck zu ersetzen.

8.8.4 Diagramm-Funktionen

Wird der Cursor im Diagramm-Fenster in den Anzeigeteil für die während der Simulation gesammelten Daten gebracht, erscheint durch Drücken der mittleren Maustaste das in Abb. 8-18 dargestellte Auswahlmenü. Die Funktionen bedeuten im einzelnen:

function
Ändert die Funktion, welche die Werte errechnet, die im Diagramm-Fenster dargestellt werden (siehe weiter vorne für eine Beschreibung der Default-Funktionen).

clear
Setzt das Diagramm zurück. Mit dieser Funktion werden die Daten, die im Diagramm-Fenster dargestellt sind, gelöscht.

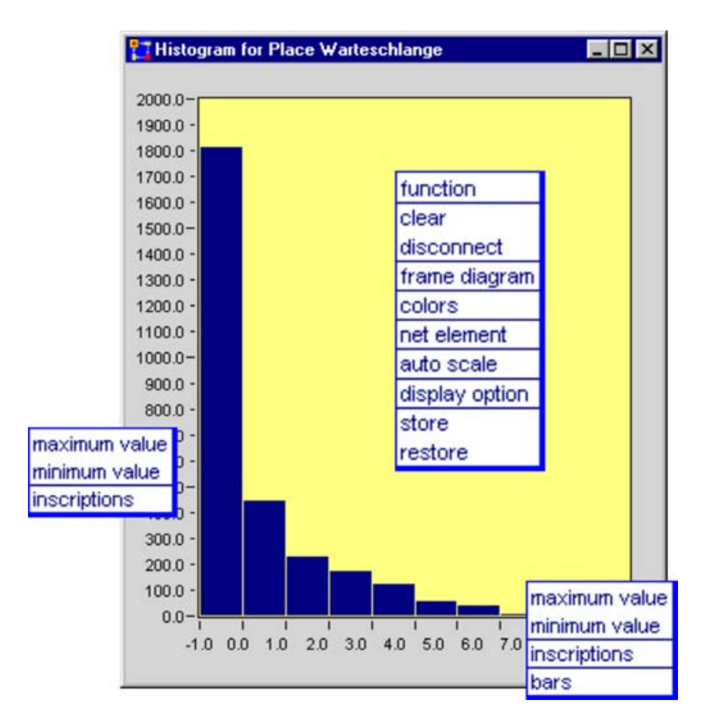


Abb. 8-18:Histogramm-Fenster mit Auswahlmenüs

disconnect
Löst das Diagramm-Fenster von dem Netzelement. Nach Ausführen dieser Funktion wird die Darstellung im abgekoppelten Diagrammfenster eingefroren. Diese Funktion ist beispielsweise sehr nützlich, um die Daten in einem Diagramm-Fenster mit denjenigen zu vergleichen, die nach der Änderung einiger System-Parameter erzeugt und in einem entsprechenden Diagramm-Fenster angezeigt werden.

frame diagram
Definiert den Fenster-Ausschnitt, in dem das Diagramm dargestellt werden soll. Diese Funktion kann beispielsweise benutzt werden,

wenn gewisse Einträge der Maßstab-Beschriftungen (Vermaßungen oder sonstiges) nicht lesbar sind.

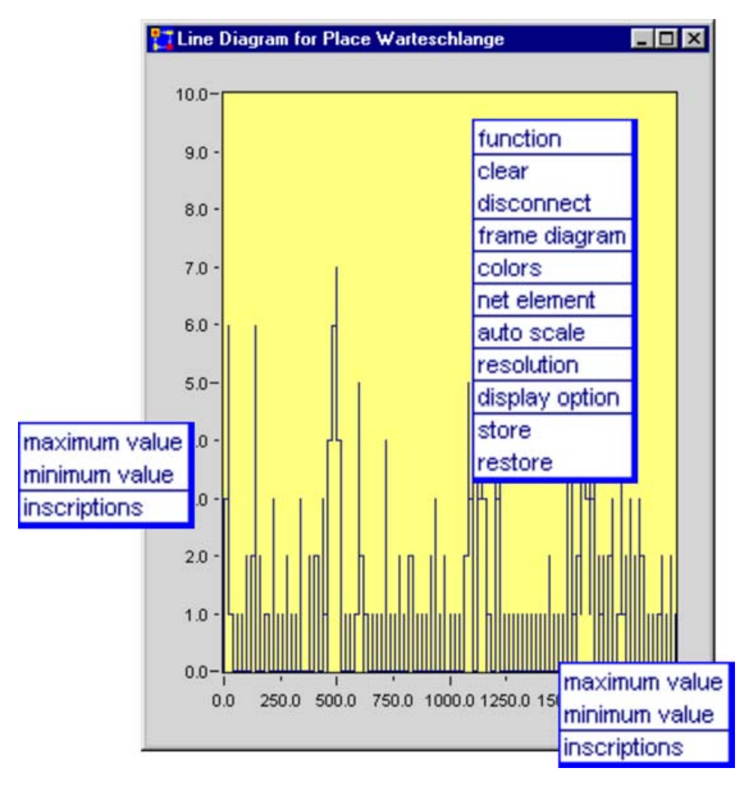


Abb. 8-19: Liniendiagramm-Fenster mit Auswahlmenüs

colors
Auswahl der Vorder- und Hintergrund-Farbe für das Ausgabefenster.

net element
Zeigt das Netz-Element an, zu dem dieses Diagramm-Fenster gehört.

auto scale
Öffnet ein Fenster mit einem Schalter, über dessen Einstellung festgelegt werden kann, ob die Maßstäbe für das Diagramm während der Simulation automatisch angepaßt werden sollen.

resolution
Diese Funktion steht nur für Liniendiagramme zur Verfügung.

Mit diesem Befehl kann die Auflösung, mit der die Linien dargestellt werden sollen, sowohl in der X- als auch in der Y-Richtung bestimmt werden. Alle Ausgaben in das Diagramm werden entsprechend gerundet.

display option
Die Funktion bearbeitet einen Schalter, der den Wert ‘on’ oder ‘off’ annehmen kann.

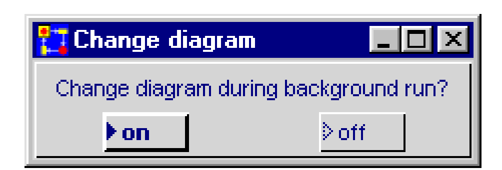


Abb. 8-20:Diagramm-Schalter

Steht der Schalter auf ‘on’ (Vorbesetzung), so werden die Diagramme auch während eines Simulationslauf im Background-Modus gezeichnet. Andernfalls werden die Diagramme erst gezeichnet, wenn der Simulationslauf unterbrochen oder beendet wird.

store
Speichert die Parameter und die Daten des Diagramm-Fensters in einem File mit frei wählbarem Namen. Das File wird im Unter-Verzeichnis 'ioutils' abgelegt und erhält die Kennung '.his' (für Balkendiagramme) oder '.lin' (für Liniendiagramme).

restore
Stellt über ein Auswahlfenster, aus dem der Benutzer ein Diagramm auswählen kann, die Parameter und die Daten eines Diagramm-Fensters wieder her. Dabei werden die Daten aus einem File mit der Kennung '.his' (für Balkendiagramme) und '.lin' (für Liniendiagramme) entnommen, das sich im Unter-Verzeichnis 'ioutils' befindet.

8.8.5 Achsen-Befehle

Wird in einem Diagramm-Fenster der Cursor zwischen einer Achsen-Abgrenzung (sowohl der X- als auch der Y-Achse) und den Fensterrand gebracht und wird danach die rechte Maustaste gedrückt, so erscheinen die in Abb. 8-18 und Abb. 8-19 links und unten dargestellten Auswahlmenüs.

maximum value
Setzt den Maximalwert des Darstellungsmaßstabs für die entsprechende Achse.

minimum value
Setzt den Minimalwert des Darstellungsmaßstabsfür die entsprechende Achse.

inscriptions
Setzt die Anzahl der Inskriptionen für die entsprechende Achse.

bars
Dieser Befehl erscheint nur bei Balkendiagrammen. Er bestimmt die Anzahl der darzustellenden Balken.

8.8.6 Balkendiagramm-Intervalle

Die obere Grenze des Intervalls der X-Achse wird bei der Errechnung des Balken berücksichtigt. Die untere Grenze des Intervalls wird dem Balken zugeschlagen, der vor dem aktuellen liegt.

8.8.7 Botschaften für Diagramme

8.8.7.1 Die Botschaft ‘resetPlaceStatistics’

Mit der Botschaft

self resetPlaceStatistics: aPlace

werden die Statistikfenster der angegebenen Stelle zurückgesetzt .

Beispiel: self resetPlaceStatistics: 'stelle1'.

8.8.7.2 Die Botschaft 'resetAllStatistics'

Mit dieser Botschaft werden sämtliche geöffneten Statistikfenster eines Netzes zurückgesetzt.

Beispiel: self resetAllStatistics.

9 ARBEITEN IM NETZ-FENSTER

In den Netz-Fenstern werden die Netze modelliert und animiert (siehe dazu auch die vorhergehende Kapitel 'Editor' und 'Simulator').

9.1 Elemente anwählen

Im folgenden wird beschrieben, wie die einzelnen Netz-Elemente angewählt und wieder abgewählt werden können. Angewählte Elemente werden mit einem kleinen schwarzen beziehungsweise weißen Quadrat markiert. Befindet sich das Netz-Fenster im Editier-Modus, ao kann eine beliebige Anzahl von Elementen gleichzeitig angewählt werden. Im Simulations-Modus ist dagegen nur jeweils ein Element anwählbar.

9.1.1 Anwählen von S- und T-Elementen

S-Elemente (Stellen und Kanäle) und T-Elemente (Transitionen und Module) können durch Drücken der linken Maustaste markiert werden, wenn sich der Cursor auf ihnen befindet.

Wird ein Netzelement von Text überdeckt, so wird durch Positionieren des Cursors auf das Netzelement und Drücken der linken Maustaste nicht das Netzelement, sondern der Text selektiert (siehe Abschitt 9.1.3). Will man das darunter liegende Netzelement markieren, so muss beim Drücken der linken Maustaste gleichzeitig die linke Shifttaste gedrückt werden.

9.1.2 Anwählen von Konnektoren

Es gibt zwei Möglichkeiten, um Konnektoren anzuwählen. Die erste besteht darin, daß man den Cursor auf den Konnektor setzt und dann die linke Maus-Taste drückt.

Ein Konnektor kann aber auch auf die gleiche Art angewählt werden, wie er erstellt wurde, nämlich durch Nachfahren des Konnektors in der entsprechenden Richtung. Dieser Mechanismus arbeitet wie ein Schalter, d.h. nochmaliges Nachfahren des Konnektors deselektiert ihn wieder.

9.1.3 Anwählen von Text

Text kann angewählt werden, indem der Cursor auf diesen positioniert, dann die linke Maus-Taste gedrückt und über den zu markierenden Textteil mit gedrückter Maustaste gestrichen wird. Es kann nur jeweils ein Text angewählt werden.

9.1.4 Anwählen von mehreren Elementen

Durch Drücken der linken Shift-Taste beim Selektiervorgang können im Editiermodus mehrere S-, T-Elemente und Konnektoren gleichzeitig angewählt werden.

9.1.5 Abwählen

Der Anwähl-Mechanismus funktioniert wie ein Schalter. Ist ein Element einmal angewählt, kann es durch Wiederholen des gleichen Vorganges wieder abgewählt werden.

Wenn sich der Cursor in dem betreffenden Netz-Fenster befindet, so können durch gleichzeitiges Drücken der Shift- und der Ctrl-Taste

alle angewählten Elemente auf einmal abgewählt werden (Bitte beachten Sie, daß auf deutschen PC-Tastaturen die Ctrl-Taste mit 'Strg' beschriftet ist.). Das gleiche erreicht man durch Drücken des ‘deselect’-Button im Netzfenster.

9.2 Elemente verschieben

9.2.1 Verschieben von S- und T-Elementen

Das Verschieben von einem oder von mehreren S- und T-Elementen kann nicht über einen Menüpunkt ausgeführt werden. Diese Operation wird folgendermaßen ausgeführt:

Zuerst müssen die S- und T-Elemente angewählt werden. Danach wird der Cursor auf eines der zu verschiebenden Elemente positioniert und die linke Maustaste niedergedrückt. Die Elemente können nun gemeinsam mit der Maus frei im Fenster bewegt werden, dabei bleibt die linke Maustaste stets niedergedrückt. Ist die gewünschte Position erreicht, können die Elemente durch Loslassen der linken Maustaste fixiert werden.

Wenn mehrere Elemente gleichzeitig verschoben werden, wird nur ein diese umspannendes Rechteck angezeigt. Wird dann während des Verschiebens zusätzlich die linke Shift-Taste gedrückt, so werden die Elemente, Konnektoren und Inskriptionen während des Verschiebungsvorgangs angezeigt.

Soll ein Netzelement, das von Text überdeckt ist, verschoben werden, so wird es zunächst wie in Abschnitt 9.1.1 beschrieben, selektiert. Bei weiterhin festgehaltener Shifttaste fängt man an, die Maus zu bewegen und lässt dann an der neuen Position die Shifttaste los.

9.2.2 Verschieben von Text

Text kann auf die gleiche Art und Weise verschoben werden, wie S- und T-Elemente. Wie beim Anwählen kann auch beim Verschieben nur mit einem einzigen Text operiert werden. Die linke Shift-Taste hat hier keinen Einfluß.

Falls ein Text nicht innerhalb eines Fensters liegt, sollte das Element, zu dem dieser Text gehört, solange verschoben werden, bis der Text wieder innerhalb des Fensters sichtbar wird. Danach kann der Text ggf. selbst an die jeweils gewünschte Stelle geschoben werden.

10 WAHRSCHEINLICHKEITS-VERTEILUNGEN

Normalerweise sind kommerzielle oder technische Prozesse in eine „Umwelt” eingebettet, mit der sie in Wechselwirkung stehen. Diese Umwelt kann häufig aus Aufwands- oder aus prinzipiellen Gründen nicht mit in das Simulationsmodell aufgenommen werden. Um das Verhalten der Umwelt trotzdem mit in das Simulationsgeschehen einbeziehen zu können, wird das Verhalten der von „außen” auf das Modell einwirkenden Prozeßgrößen, sofern es sich nicht algorithmisch darstellen läßt, über empirische oder mathematische Verteilungsfunktionen approximiert. Lassen wir ein Simulationsmodell nur hinreichend lange ablaufen, so erhalten wir auf diese Weise repräsentative Aussagen über das Verhalten des modellierten Systems.

Will man die Auslegung von Resourcen durch Simulation optimieren, so muß man zunächst die Verteilungsfunktion für die relevanten statistisch verteilten Modellgrößen bestimmen und deren Verteilungen dann in das Modell einbauen. In den folgenden Abschnitten wird der Aufruf von mathematischen Verteilungsfunktionen beschrieben, die häufig vorkommen und die deshalb standardmäßig in PACE vorgesehen sind. Da die Auswahl der richtigen Verteilungsfunktion für die Verwendbarkeit der Simulationsergebnisse von großer Bedeutung ist, werden bei verschiedenen Verteilungsfunktionen Hinweise auf mögliche Einsatzfelder gegeben. In den Fällen, in denen die Verteilung der Eingangsgrößen nicht bekannt ist, können aus Messdaten empirische Verteilungsfunktionen erzeugt werden.

Das Zusammenspiel zwischen Petri-Netzen und Verteilungsfunktionen wird an zwei Beispielen dargelegt. In einem weiteren Beispiel wird die Erzeugung und Verwendung einer empirischen Verteilung demonstriert.

10.1 Diskrete Verteilungen

Diskrete Verteilungsfunktionen werden für Zufallsgrößen verwendet, die nur eine endliche oder eine abzählbar unendliche Anzahl von Werten annehmen können.

10.1.1 Bernoulli-Verteilung

Die Bernoulli-Verteilung wird verwendet, wenn eine Zufallsgröße nur zwei Werte, jeden davon mit einer bestimmten Wahrscheinlichkeit, annehmen kann. Sie liefert Werte für das Eintreten eines der beiden Ereignisse beim nächsten Versuch..

Klassenname: Bernoulli

Meldungen:

parameter: x
: x ist die Wahrscheinlichkeit für das Auftreten einer der beiden Möglichkeiten.

parameter: x seed: seedNumber
: x ist die Wahrscheinlichkeit für das Auftreten einer der beiden Möglichkeiten. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

parameter: x use: aRandom
: während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

parameter: x use: aRandom seed: seedNumber
wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung, also entweder 0 oder 1.

mean — liefert den Parameter einer Bernoulli-Verteilung

variance — liefert die Varianz einer Bernoulli-Verteilung

density: x — liefert die Wahrscheinlichkeiten für Erfolg (x=1) und Mißerfolg (x=0)

Anwendungen: Neben den klassischen Anwendungen: (Münzenwurf, Würfeln, Ziehen einer Spielkarte usw.) kann die Verteilung für die Beschreibung der Verfügbarkeit einer Resource verwendet werden.

Beispiel:

```
(Bernoulli parameter: 0.4) next
```

oder

```
bern := Bernoulli parameter: 0.4
                  seedNumber: 713.
bern next
```

liefern jeweils einen Wert dafür, daß eine Resource, die im Mittel in 4 von 10 Fällen vorgefunden wird, verfügbar ist.

10.1.2 Binomial-Verteilung

Die Binomial-Verteilung erweitert die Bernoulli-Verteilung und beantwortet die Frage, wie oft ein Ereignis in den nächsten n Versuchen eintritt.

Klassenname: Binomial

Meldungen:

events: n mean: x

definiert eine Verteilung für eine Anzahl von Versuchen n mit einem Erwartungswert x für die Zufallsgröße.

events: n mean: x seed: seedNumber

definiert eine Verteilung für eine Anzahl von Versuchen n mit einem Erwartungswert x für die Zufallsgröße. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

events: n mean: x use: aRandom

während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

events: n mean: x use: aRandom seed: seedNumber

wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — ergibt den nächsten Wert der Verteilung.

nean — liefert den Erwartungswert einer Verteilung.

variance — liefert die Varianz einer Verteilung

density: q ergibt die Wahrschenlichkeit dafür, daß ein Ereignis q mal eintritt.

Anwendung: Verfügbarkeit von mehreren Resourcen**.**

Beispiel: Für die Verteilung

bino := (Binomial events: 6 mean: 3)

liefert

bino next

einen Wert zwischen 1 und 6, der angibt, wieviele Resourcen verfügbar sind, wenn im Mittel 3 Resourcen vorhanden sind.

10.1.3 Geometrische Verteilung

Die geometrische Verteilung erweitert die Bernoulli-Verteilung und beantwortet die Frage, wieviele Versuche durchgeführt werden müssen, bevor ein Ereignis eintritt.

Klassenname: Geometric

Methoden:

mean: x definiert eine geometrische Verteilung mit Erwartungswert x für die Zufallsgröße.

mean: x seed: seedNumber
definiert eine geometrische Verteilung mit Erwartungswert x für die Zufallsgröße. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

mean: x use: aRandom
während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

mean: x use: aRandom seed: seedNumber
wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next liefert den nächsten Wert der Verteilung.

mean liefert den Erwartungswert der Verteilung.

variance liefert die Varianz der Verteilung.

density: n liefert die Wahrscheinlichkeit dafür, daß n Ereignisse eintreten, bevor ein bestimmtes Ereignis eintritt.

Anwendung: Zeitspanne bis das nächste Werkstück, der nächste Patient usw. erscheint.

Beispiel: Wenn in der Minute 3 Werkstücke eintreffen, so können wir die Verteilung wie folgt angeben:

ankunft := Geometric mean: 3.

Der nächste Wert der Verteilung:

ankunft next

gibt an, nach wievielen Minuten das nächste Werkstück erscheint.

Die Wahrscheinlichkeit, daß das nächste Werkstück nach 10 Minuten eintrifft, ergibt sich aus:

ankunft density: 10.

10.1.4 Poisson-Verteilung

Die Poisson-Verteilung beantwortet die Frage nach der Wahrscheinlichkeit für das mehrfache Auftreten eines Ereignises.

Klassenname: Poisson

Methoden:

mean: x
: definiert eine Poisson-Verteilung mit Erwartungswert x.

mean: x seed: seedNumber
: definiert eine Poisson-Verteilung mit Erwartungswert x. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

mean: x use: aRandom
: während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

mean: x use: aRandom seed: seedNumber
: wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next
: liefert den nächsten Wert der Verteilung.

mean	liefert den Erwartungswert der Verteilung.
variance	liefert die Varainz der Verteilung.
density: x	ergibt die Wahrscheinlichkeit dafür, daß x Ereignisse gleichzeitig auftreten.

Anwendung: Die Poisson-Verteilung ist eine gute Näherung für die Verteilung von Kundenanfragen, wie sie z.B. bei Dienstleistungen und im technischen Service vorkommen.

Beispiel: Wenn in der Stunde im Mittel 10 Anfragen eintreffen, so lautet die zugehörige Poisson-Verteilung:

```
anfragen := Poisson mean:10.
```

Der Wert:

```
anfragen next
```

liefert einen Wert für die Anzahl der Anfragen, die in der nächsten Stunde eintreffen werden.

Über den Aufruf:

```
anfragen density: n
```

erhalten wir die Wahrscheilichkeit, daß in einer Stunde n Anfragen eintreffen werden.

10.2 Kontinuierliche Verteilungen

Kontinuierliche Verteilungsfunktionen werden für Zufallsgrößen verwendet, die jeden beliebigen Wert in einem oder mehreren Intervallen annehmen können. Bei Simulationen werden sie meist für die Modellierung von Zeitereignissen eingesetzt.

10.2.1 Gleichverteilung

Eine Zufallsgröße ist gleichverteilt, wenn sie jeden Wert in einem vorgegebenen Intervall mit gleicher Wahrscheinlichkeit annehmen kann.

Dichte der Verteilung: $\frac{1}{b-a}$ für a <= x <= b

0 sonst

Klassenname: Uniform

Methoden:

from: x to: y
: erzeugt eine Gleichverteilung mit der unteren Grenze x und der oberen Grenze y.

from: x to: y seed: seedNumber
: erzeugt eine Gleichverteilung mit der unteren Grenze x und der oberen Grenze y. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

from: x to: y use: aRandom
: während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator

verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

from: x to: y use: aRandom seed: seedNumber
wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Mittelwert der Verteilung.

variance — liefert die Varianz der Verteilung.

Anwendung: unbedingte Zufallsgrößen

10.2.2 Exponentialverteilung

Die Exponentialverteilung wird dazu verwendet, um die Zeitspanne bis zum Eintritt des nächsten Ereignisses zu bestimmen.

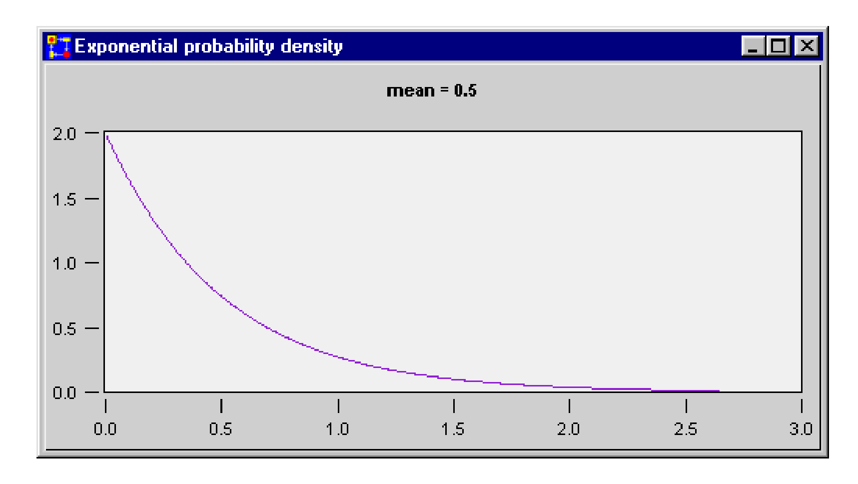


Abb. 10-1: Exponentialverteilung

Dichte der Verteilung: $\frac{1}{\beta} e^{-x/\beta}$ für x > 0

0 sonst

Klassenname: Exponential

Methoden:

mean: x
: erzeugt eine Exponentialverteilung mit Erwartungswert x.

mean: x seed: seedNumber
: erzeugt eine Exponentialverteilung mit Erwartungswert x. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber

eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

mean: x use: aRandom

während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

mean: x use: aRandom seed: seedNumber

wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Mittelwert der Verteilung.

variance — liefert die Varianz der Verteilung.

density: x — liefert die Wahrscheinlichkeit dafür, daß die Zeitspanne kleiner als x ist.

Anwendung: Die Exponentialverteilung liefert für Poisson-verteilte Zustandgrößen die Zeitspanne bis zum Eintreffen des nächsten Ereignisses. Sie kann auch für Systeme eingesetzt werden, die einer natürlichen Alterung unterliegen (Glühlampen, Elekronik-Bausteine, usw.).

Beispiel: (Exponential mean: 10) next

liefert den nächsten Wert einer exponentiell verteilten Zufallsgröße mit Mittelwert 10.

10.2.3 Cauchy-Verteilung

Die Cauchy-Verteilungsfunktion ist in der Physik unter dem Namen Resonanzkurve bekannt.

Dichte der Verteilung: $\frac{\beta}{\pi((x-\mu)^2+\beta^2)}$ $\quad -\infty < \mu < +\infty$

$0 < \beta < +\infty$

μ wird als shape-Parameter, β als scale-Parameter bezeichnet.

Klassenname: Cauchy

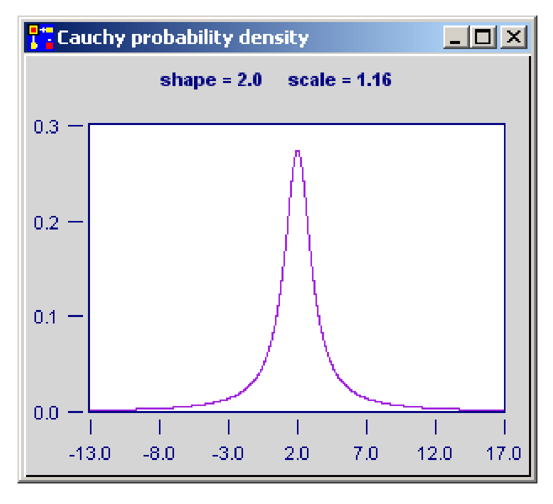


Abb. 10-2: Cauchy-Verteilung

Verteilungsfunktion: $\frac{1}{2} + \frac{1}{\pi} \arctan(\frac{x-\mu}{\beta})$

Methoden:

shape: mu scale: beta
: erzeugt eine Cauchy-Verteilung mit Parametern mu und beta.

shape: mu scale: beta seed: seedNumber
: erzeugt eine Cauchy-Verteilung mit Parametern mu und beta. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

shape: mu scale: beta use: aRandom
: während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

shape: mu scale: beta use: aRandom seed: seedNumber
: wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next
: liefert den nächsten Wert der Verteilung.

mean
: liefert den shape-Parameter.

variance
: liefert den Wert nil. Die Varianz einer Cauchy-Verteilung ist nicht definiert.

density: x
: liefert den Wert der Wahrscheinlichkeitsdichte für an der Stelle x.

distributionValue: x

liefert den Wert der Verteilungsfunktion für das Argument x.

10.2.4 Gamma-Verteilung

Spezialfälle der Gamma-Verteilung sind u.a. die Chi-Square-Verteilung, die Erlang-Verteilung und die Exponentialverteilung.

Dichte der Verteilung: $\dfrac{x^{\alpha-1} e^{-x/\beta}}{\beta^{\alpha}\Gamma(\alpha)}$ für x > 0

0 so nst

Klassenname: Gamma

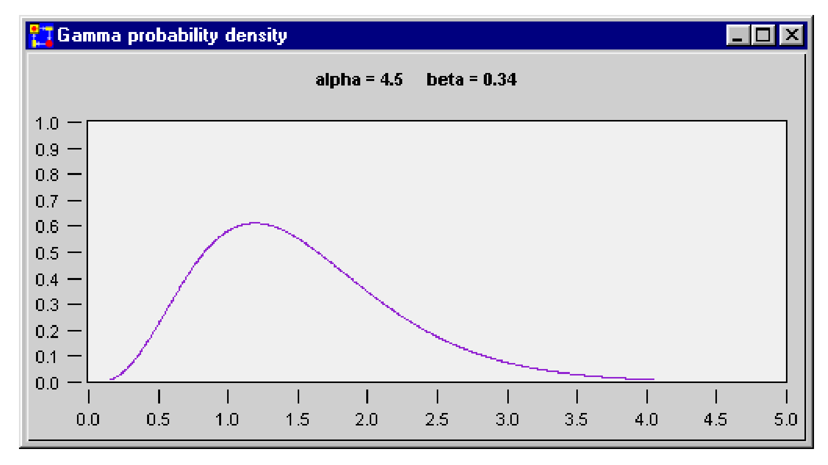


Abb. 10-2: Gamma-Verteilung

Methoden:

alpha: a beta: b
: erzeugt eine Gamma-Verteilung für die Parameter a > 0 und b > 0.

alpha: a beta: b seed: seedNumber
: erzeugt eine Gamma-Verteilung für die Parameter a > 0 und b > 0. seedNumber ist oben beschrieben.

alpha: a beta: b use: aRandom
: während bei den beiden vorangegangenen

Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

alpha: a beta: b use: aRandom seed: seedNumber
: wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next
: liefert den nächsten Wert der Verteilung.

mean
: liefert den Erwartungswert der Verteilung.

variance
: liefert die Varianz der Verteilung.

density: x
: liefert die Wahrscheinlichkeistsdichte an der Stelle x.

10.2.5 Erlang-Verteilung

Die Erlang-Verteilung beantwortet die Frage, wieviel Zeit bis zum Auftreten des n-ten Ereignisses vergeht. Sie wird in der Warteschlangentheorie zur Beschreibung von zufälligen Bedienzeiten und von Ankunftszeiten aufeinanderfolgender Ereignisse verwendet.

Die Erlang-Verteilung ist ein Spezialfall der Gamma-Verteilung für natürliche Zahlen α.

Dichte der Verteilung: $\frac{x^{n-1} e^{-\frac{x}{\beta}}}{\beta^n (n-1)!}$ für x >= 0

0 für x < 0

Da der Erwartungswert der Erlang-Verteilung p = n . β ist, kann man dafür auch schreiben:

$$\left(\frac{n}{p}\right)^n \frac{x^{n-1} e^{-\frac{nx}{p}}}{(n-1)!}$$ für x >= 0

0 für x < 0.

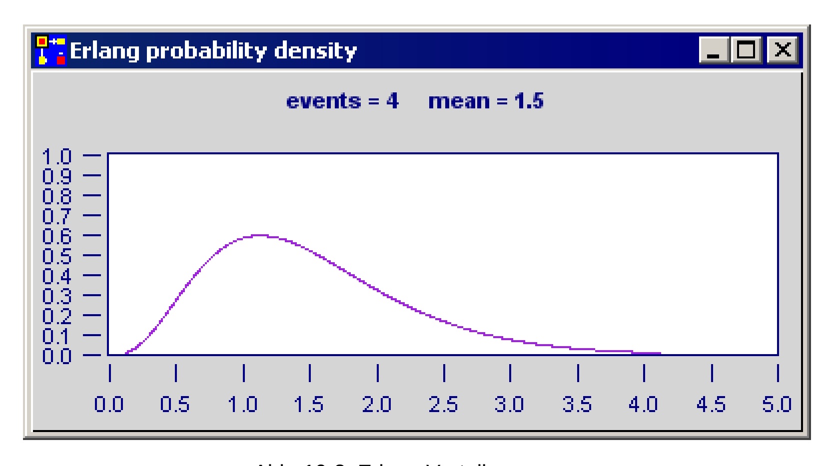


Abb. 10-3: Erlang-Verteilung

Klassenname: Erlang

Methoden:

events: n mean: p
erzeugt eine Erlang-Verteilung für n Ereignisse mit Mittelwert p. n ist eine natürliche ganze Zahl.

events: n mean: p seed: seedNumber
erzeugt eine Erlang-Verteilung für n Ereignisse mit Mittelwert p. n ist eine natürliche ganze Zahl. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

events: n mean: p use: aRandom
während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

events: n mean: p use: aRandom seed: seedNumber
wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Erwartungswert der Verteilung.

variance — liefert die Varianz der Verteilung.

density: x — liefert die Wahrscheinlichkeit dafür, daß die Zeitspanne bis zum Eintritt des n-ten Ereignisses kleiner als x ist.

Anwendung: Planung von Lagerbeständen.

Beispiele: Wieviele Bauteile werden im nächsten Monat ausfallen?

Für ein Bauteil wurde eine MTBF von 2 Monaten (60 Tage) festgestellt. Die Verteilung:

```
ausfall := Erlang events: 1000 mean: 60
```

liefert die Verteilung für 1000 Ausfälle des Bauteils. Die Wahrscheinlichkeit, daß 1000 Ausfälle schon nach einem Monat auftreten ist:

```
ausfall density: 30
```

10.2.6 Chi-Quadrat-Verteilung

Mit der Chi-Quadrat-Verteilung, die ein Spezialfall der Gamma-Verteilung ist, kann man die Frage zu beantworten, wie gut man mit einer bestimmten Theorie die beobachteten Messwerte berechnen kann, oder mit anderen Worten, wie gut die verwendete Theorie ist. Sie kann deshalb z.B. eingesetzt werden, wenn man die Güte eines Modells an der Realität messen will.

Dichte der Verteilung: $\frac{x^{\frac{n}{2}-1}e^{-\frac{x}{2}}}{2^{\frac{n}{2}}\Gamma(\frac{n}{2})}$ n eine positive ganze Zahl

Klassenname: ChiSquare

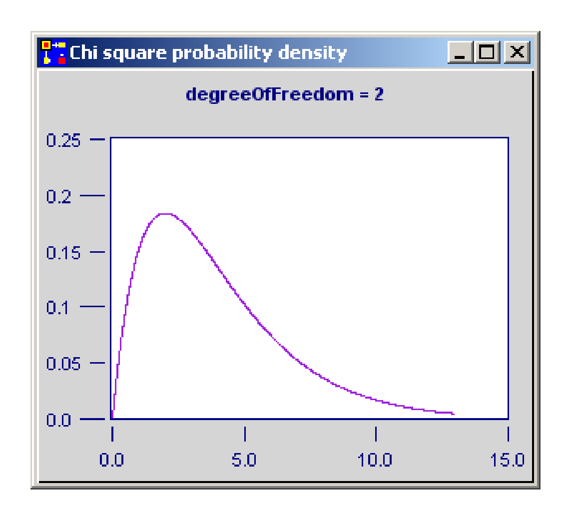


Abb. 10-4: Chi-Quadrat-Verteilung

Methoden:

degreeOfFreedom: n

erzeugt eine Chi-Quadrat-Verteilung mit Freiheitsgrad n. n ist eine natürliche ganze Zahl.

degreeOfFreedom: n seed: seedNumber

erzeugt eine Chi-Quadrat-Verteilung mit Freiheitsgrad n. n ist eine natürliche ganze Zahl. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

degreeOfFreedom: n use: aRandom

während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

degreeOfFreedom: n use: aRandom seed: seedNumber

wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Mittelwert der Verteilung (= n).

variance — liefert die Varianz der Verteilung (= 2n)

density: x — Wahrscheinlichkeitsdichte an der Stelle x.

10.2.7 Normal-Verteilung

Die Kenntnis der Normal- oder Gauß-Verteilung gehört für jeden Entwickler zur Allgemeinbildung. Sie wurde vom jugendlichen Gauß 1794 bei seinen Arbeiten über Meßfehler gefunden.

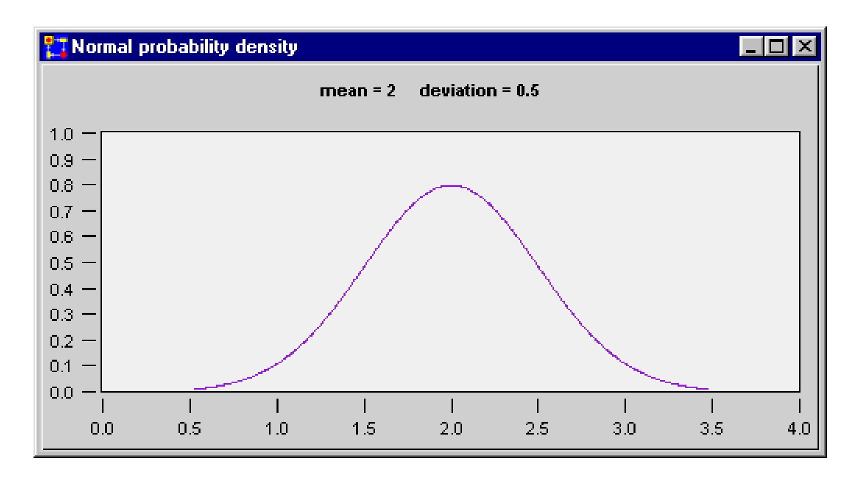


Abb. 10-5: Normal- oder Gauß-Verteilung

Dichte der Verteilung: $$\frac{1}{\sqrt{2\pi\sigma^2}}\mathrm{e}^{-(x-\mu)^2/2\sigma^2}$$

Klassenname: Normal

Methoden:

mean: x deviation: y

erzeugt eine Gauß-Verteilung um den Mittelwert x mit Standard-Abweichung y.

mean: x deviation: y seed: seedNumber

erzeugt eine Gauß-Verteilung um den Mittelwert x mit Standard-Abweichung y. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare

Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

mean: x deviation: y use: aRandom
während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

mean: x deviation: y use: aRandom seed: seedNumber
wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Gauß-Verteilung.

mean — liefert den Mittelwert der Verteilung

variance — liefert die Varianz der Verteilung

density: x — ergibt die Wahrscheinlichkeit dafür, daß der nächste Wert der Zufallsgröße im Intervall zwischen -x und +x liegt.

Anwendung: bei vielen Anwendungen (insbesondere bei der Berücksichtigung von Beobachtungsfehlern).

Die Gauß-Verteilung kann häufig in der Umgebung des Scheitelpukts zur Approximation anderer Verteilungen verwendet werden, bei denen es einen zentralen Mittelwert gibt, von dem aus die Verteilung nach beiden Seiten hin stark abfällt.

Beispiel: gauss := Normal mean: 10 deviation 2.

10.2.8 Fisher-Tippett-Verteilung

Dichte der Verteilung: $\frac{1}{\beta}e^{-\frac{x-\alpha}{\beta}-e^{\frac{x-\alpha}{\beta}}}$ $\quad -\infty < \alpha < +\infty$

$0 < \beta < +\infty$

Klassenname: FisherTippett

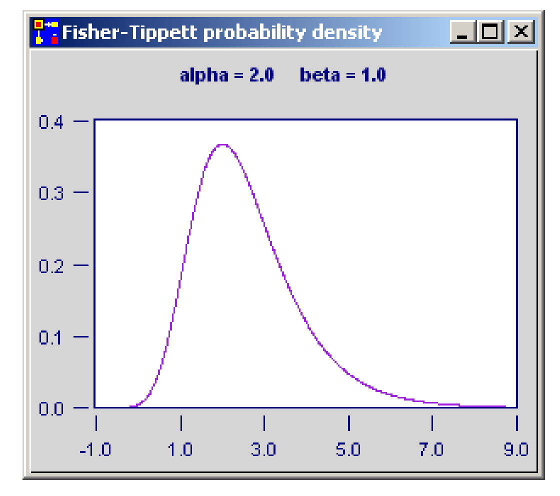


Abb. 10-6:Fisher-Tippett-Verteilung

Verteilungsfunktion: $e^{-e^{-\frac{x-\alpha}{\beta}}}$

Mittelwert: $\alpha + \gamma\beta$

γ = 0.57721... ist die Eulersche Konstante

Varianz: $\frac{\pi\beta}{\sqrt{6}}$

Methoden:

alpha: a beta: b
erzeugt eine Fisher-Tippett-Verteilung.

alpha: a beta: b seed: seedNumber
erzeugt eine Fisher-Tippett-Verteilung. seedNumber ist oben beschrieben.

alpha: a beta: b use: aRandom
während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

alpha: a beta: b use: aRandom seed: seedNumber
wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next
liefert den nächsten Wert der Verteilung.

mean
liefert den Erwartungswert der Verteilung.

variance
liefert die Varianz der Verteilung.

density: x
liefert die Wahrscheinlichkeit dafür, daß die Zeitspanne bis zum Eintritt des n-ten Ereignisses kleiner als x ist.

10.2.9 Weibull-Verteilung

Die Weibull-Verteilung hat eine ähnliche Form wie die Gamma-Verteilung.

Dichte der Verteilung: $\alpha\beta^{-\alpha}x^{\alpha-1}e^{-[x/\beta]^{\alpha}}$ für x > 0

0 sonst

Klassenname: Weibull

Methoden:

shape: alpha scale: beta

erzeugt eine Weibull-Verteilung.

shape: alpha scale: beta seed: seedNumber

erzeugt eine Weibull-Verteilung. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

shape: alpha scale: beta use: aRandom

während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

shape: alpha scale: beta use: aRandom seed: seedNumber

wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next	liefert den nächsten Wert der Verteilung.
mean	liefert den Erwartungswert der Verteilung.
variance	liefert die Varianz der Verteilung.
density: x	liefert die Wahrscheinlichkeitsdichte an der Stelle x.

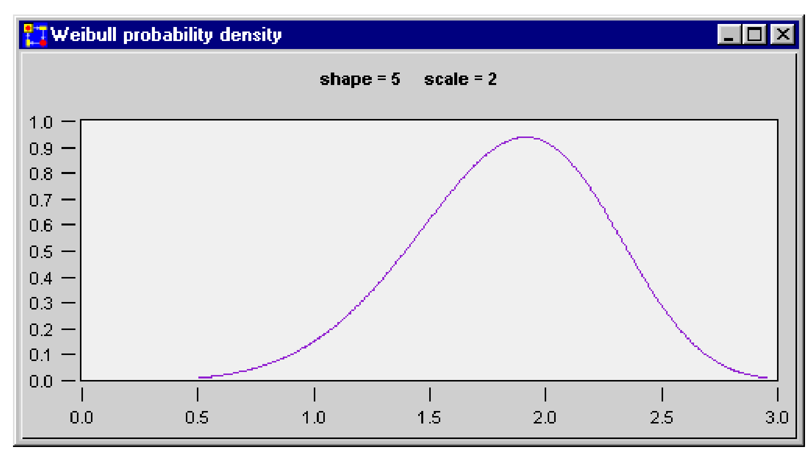


Abb. 10-7: Weibull-Verteilung

Anwendung: Die Verteilung wird für die Erzeugung von Zufallsgrößen verwendet, mit denen Fertigstellungs- und Ausfallzeiten beschrieben werden.

Beispiel: Weibull shape: 3 scale: 0.5.

10.2.10 Dreieck-Verteilung

Dichte der Verteilung: $\frac{2(x-a)}{(b-a)(c-a)}$ für a <= x <= c

$\frac{2(b-x)}{(b-a)(b-c)}$ für c < x <= b

0 sonst

Klassenname: Triangular

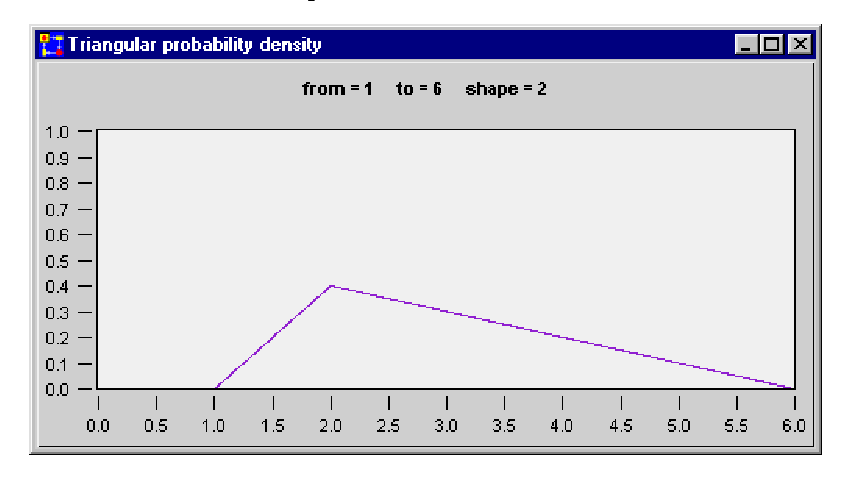


Abb. 10-8: Dreieck-Verteilung

Methoden:

from: a to: b shape: c
erzeugt eine Dreieck-Verteilung.

from: a to: b shape: c seed: seedNumber
erzeugt eine Dreieck-Verteilung. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze

Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

from: a to: b shape: c use: aRandom
: während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

from: a to: b shape: c use: aRandom seed: seedNumber
: wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Erwartungswert der Verteilung.

variance — liefert die Varianz der Verteilung.

density: x — liefert die Wahrscheinlichkeitsdichte an der Stelle x.

Anwendung: Die Dreieck-Verteilung wird häufig als erstes oder Rohmodell verwendet, wenn nur ungenügende Daten über die statistische Verteilung der zu beschreibenden Größe vorliegen.

Beispiel: Triangular from: 1.5 to: 3.7 shape: 3

10.2.11 Beta-Verteilung

Die Beta-Verteilung wird als erste rohe Näherung verwendet, wenn nur ungenügende Daten über den genauen Verlauf einer Verteilung vorliegen.

Sie wird häufig zur statistischen Beschreibung der Aufteilung von Objektmengen verwendet. Beispielsweise kann sie bei der Modellierung der Zerlegung einer Sendung in fehlerfreie und fehlerhafte Güter eingesetzt werden. Die Verteilung wird auch zur Beschreibung von Fertigstellungszeiten (PERT) eingesetzt.

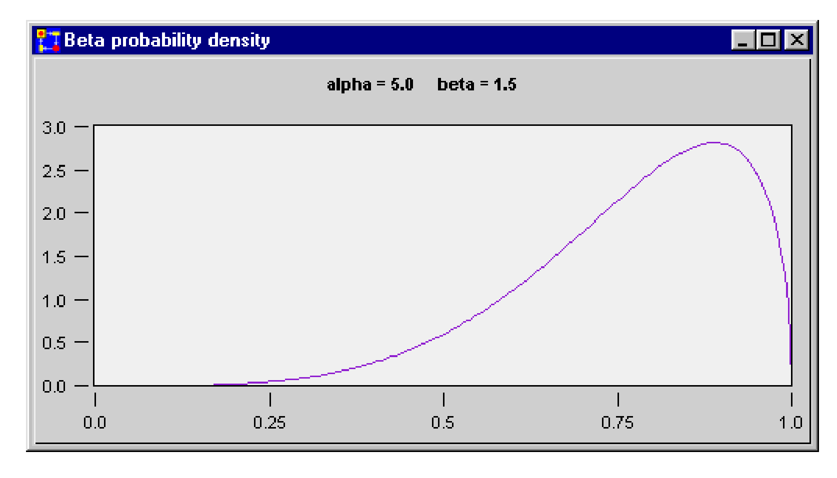


Abb. 10-9: Beta-Verteilung

Dichte der Verteilung: $\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$ für 0 < x <1

0 sonst

$B(\alpha,)\beta)$ Beta-Funktion (siehe Pearson Type 6 -Verteilung)

Klassenname: Beta

Methoden:

alpha: a beta: b

erzeugt eine Beta-Verteilung mit Form-Parametern a > 0 und b > 0.

alpha: a beta: b seed: seedNumber

erzeugt eine Beta-Verteilung mit Form-Parametern a > 0 und b > 0. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

alpha: a beta: b use: aRandom1 use: aRandom2

während bei den beiden vorangegangenen Methoden zwei Default-Zufallszahlengeneratoren verwendet werden, kann der Anwender mit dieser Methode zwei eigene, voneinander unabhängige Zufallsgenerator vorgeben. Dieser werden, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

alpha: a beta: b use: aRandom1 use: aRandom2 seed: seedNumber

wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Erwartungswert der Verteilung.

variance — liefert die Varianz der Verteilung.

density: x — liefert die Wahrscheinlichkeitsdichte an der Stelle x.

Anwendung: Die Verteilung wird für die Erzeugung von Zufallsgrößen verwendet, mit denen Fertigstellungs- und Ausfallzeiten beschrieben werden.

Beispiel: Beta alpha: 3 beta: 0.5

10.2.12 Laplace-Verteilung

Dichte der Verteilung: $\frac{e^{-\frac{|x-a|}{\beta}}}{2\beta}$ $-\infty < a < +\infty$

$0 < \beta < +\infty$

Klassenname: Laplace

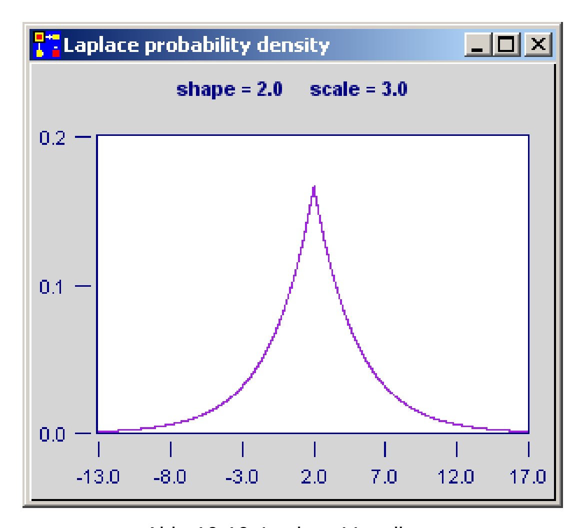


Abb. 10-10: Laplace-Verteilung

Mittelwert: $a+\beta$

Varianz: $2\beta^2$

Methoden:

shape: alpha scale: beta
erzeugt eine Laplace-Verteilung.

shape: alpha scale: beta seed: seedNumber
erzeugt eine Laplace-Verteilung. seedNumber ist

eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

shape: alpha scale: beta use: aRandom

während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

shape: alpha scale: beta use: aRandom seed: seedNumber

wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Erwartungswert der Verteilung.

variance — liefert die Varianz der Verteilung.

density: x — liefert die Wahrscheinlichkeitsdichte an der Stelle x.

10.2.13 Logistische Verteilung

Dichte der Verteilung: $\frac{\pi}{\sigma\sqrt{3}} \frac{e^{-\frac{\pi}{\sigma\sqrt{3}}(x-\mu)}}{(1+e^{-\frac{\pi}{\sigma\sqrt{3}}(x-\mu)})^2}$

Klassenname: LogisticalDistribution

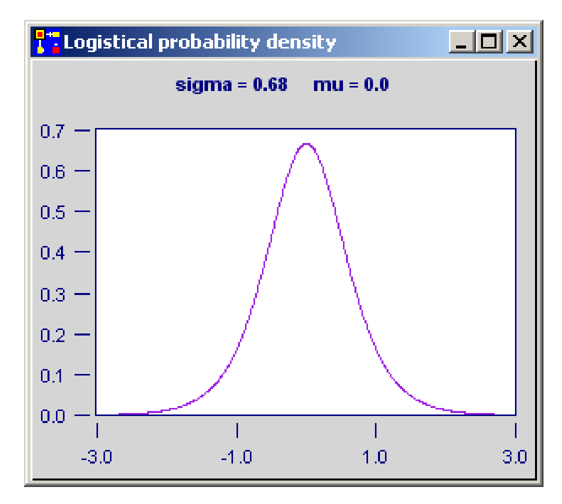


Abb. 10-11: Logistische Verteilung

Mittelwert: μ

Varianz: σ^2

Methoden:

sigma: sigmaNumb mu: muNumb
erzeugt eine logistische Verteilung.

sigma: sigmaNumb mu: muNumb seed: seedNumber

erzeugt eine .logistische Verteilung. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

sigma: sigmaNumb mu: muNumb use: aRandom

während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

sigma: sigmaNumb mu: muNumb aRandom seed: seedNumber

wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Erwartungswert der Verteilung.

variance — liefert die Varianz der Verteilung.

density: x — liefert die Wahrscheinlichkeitsdichte an der Stelle x.

10.2.14 LogNormal-Verteilung

Die LogNormal-Verteilung hat eine ähnliche Form wie die Gamma- und Weibull-Verteilung für alpha > 1. Sie kann aber in der Nähe des Koordinaten-Nullpunkts eine scharfe Spitze haben, was gelegentlich nützlich ist.

Dichte der Verteilung: $\frac{1}{x\sqrt{2\pi\sigma^2}} \exp \frac{-(\ln x - \mu)^2}{2\sigma^2}$ für x > 0

0 sonst

Klassenname: LogNormal

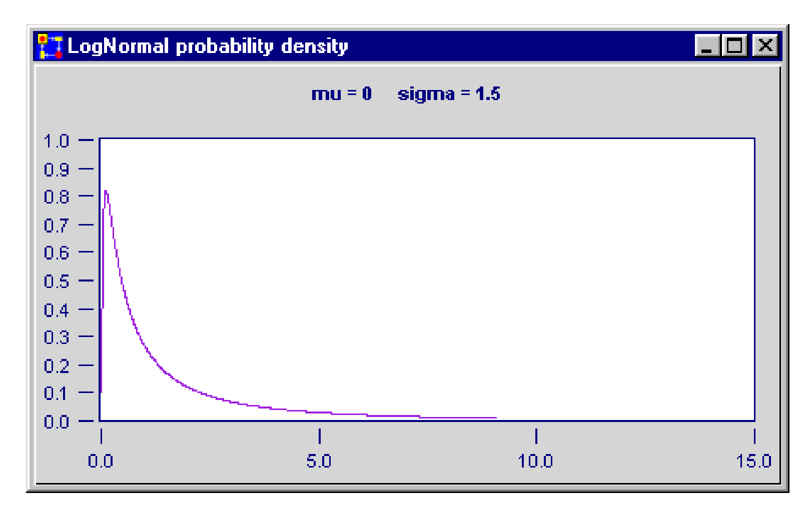


Abb. 10-12: Beispiel für eine LogNormal-Verteilung

Methoden:

mu: a sigma: b

erzeugt eine LogNormal-Verteilung mit Skalierungs-Parameter a und Form-Parameter b > 0.

mu: a sigma: b seed: seedNumber

erzeugt eine LogNormal-Verteilung mit Skalierungs-Parameter a und Form-Parameter b > 0. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

mu: a sigma: b use: aRandom

während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

mu: a sigma: b use: aRandom seed: seedNumber

wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Erwartungswert der Verteilung.

variance — liefert die Varianz der Verteilung.

density: x — liefert die Wahrscheinlichkeitsdichte an der Stelle x.

Anwendung: Die Verteilung wird für die Erzeugung von Zufallsgrößen verwendet, mit denen Bearbeitungszeiten beschrieben werden.

Beispiel: LogNormal mu: 3 sigma: 0.5

10.2.15 Pearson Type 5 -Verteilung

Diese Verteilung kann bei geeigneter Parametrierung wie die LogNormal-Verteilung in der Nähe des Ursprungs eine scharfe Spitze tragen, die höher als bei der LogNormal-Verteilung ausfallen kann.

Dichte der Verteilung: $\frac{x^{-(\alpha+1)} e^{-\beta/x}}{\beta^{-\alpha}\Gamma(\alpha)}$ für x > 0

0 sonst

Klassenname: PearsonType5

Methoden:

alpha: a beta: b

erzeugt eine Pearson Type 5-Verteilung mit Form-Parameter a > 0 und Skalierungs-Parameter b > 0.

alpha: a beta: b seed: seedNumber

erzeugt eine Pearson Type 5-Verteilung mit Form-Parameter a > 0 und Skalierungs-Parameter b > 0. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

alpha: a beta: b use: aRandom

während bei den beiden vorangegangenen Methoden ein Default-Zufallszahlengenerator verwendet wird, kann der Anwender mit dieser Methode einen eigenen Zufallsgenerator

vorgeben. Dieser wird, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

alpha: a beta: b use: aRandom seed: seedNumber
: wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next
: liefert den nächsten Wert der Verteilung.

mean
: liefert den Erwartungswert der Verteilung. Dieser existiert nur für a > 1.

variance
: liefert die Varianz der Verteilung. Diese existiert nur für a > 2.

density: x
: liefert die Wahrscheinlichkeitsdichte an der Stelle x.

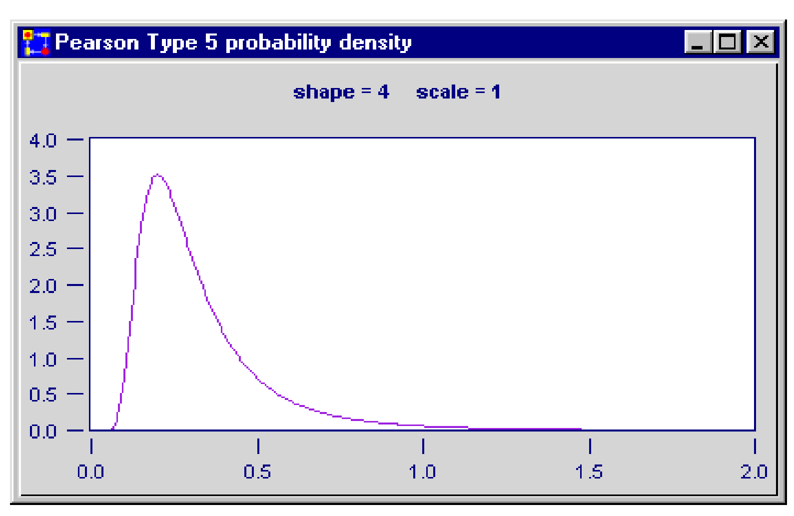


Abb. 10-13: PearsonType 5 -Verteilung

Anwendung: Die Verteilung wird für die Erzeugung von Zufallsgrößen verwendet, mit denen Fertigstellungs- und Ausfallzeiten beschrieben werden.

Beispiel: PearsonType5 alpha: 4 beta: 1

10.2.16 Pearson Type 6 -Verteilung

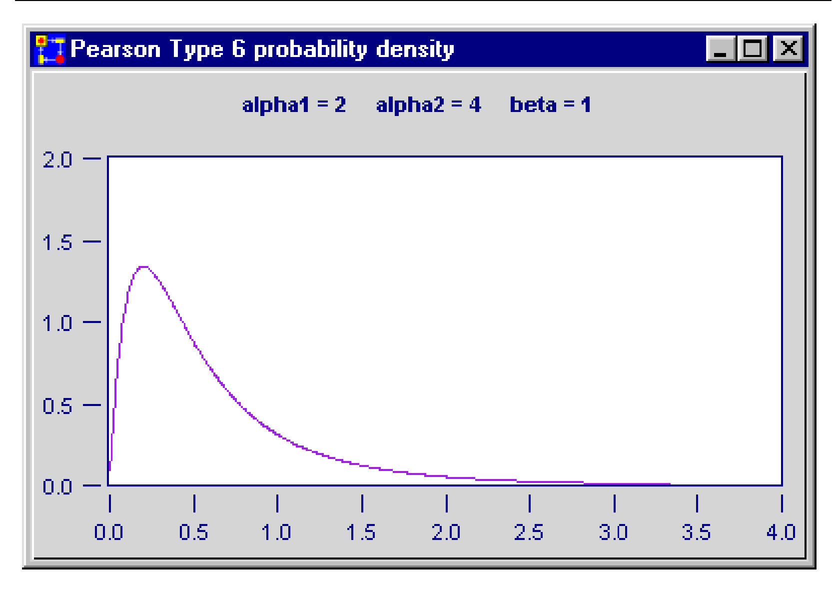


Abb. 10-14: Pearson Type 6 -Verteilung

Dichte der Verteilung: $$\frac{(x/\beta)^{\alpha_1 - 1}}{\beta\, B(\alpha_1,\alpha_2)\,(1 + (x/\beta))^{\alpha_1 + \alpha_2}}$$

für x > 0 sonst 0.

Darin ist B die Beta-Funktion:

$$B(\alpha_1,\alpha_2) = \frac{\Gamma(\alpha_1)\,\Gamma(\alpha_2)}{\Gamma(\alpha_1 + \alpha_2)}$$

Klassenname: PearsonType6

Methoden:

alpha1: a alpha2: b beta: c

erzeugt eine Pearson Type 6-Verteilung mit Form-Parametern a > 0, b > 0 und Skalierungs-Parameter c > 0.

alpha1: a alpha2: b beta: c seed: seedNumber

erzeugt eine Pearson Type 6-Verteilung mit Form-Parametern a > 0, b > 0 und Skalierungs-Parameter c > 0. seedNumber ist eine positive ganze Zahl oder 0. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

alpha1: a alpha2: b beta: c use: aRandom1 use: aRandom2

während bei den beiden vorangegangenen Methoden zwei Default-Zufallszahlengeneratoren verwendet werden, kann der Anwender mit dieser Methode zwei eigene, voneinander unabhängige Zufallsgenerator vorgeben. Dieser werden, wie in der PACE-Smalltalk-Fibel beschrieben, mit dem Klasse Random erzeugt.

alpha1: a alpha2: b beta: c use: aRandom1 use: aRandom2 seed: seedNumber

wie die vorangehend beschriebene Methode. Der Anwender kann eine seedNumber > 0 vorgeben und damit u.a. reproduzierbare Ströme von Zufallszahlen erzeugen.

next — liefert den nächsten Wert der Verteilung.

mean — liefert den Erwartungswert der Verteilung. Dieser existiert nur für b > 1.

variance	liefert die Varianz der Verteilung. Diese existiert nur für $b > 2$.
density: x	liefert die Wahrscheinlichkeitsdichte an der Stelle x.
Anwendung:	Die Verteilung wird für die Erzeugung von Zufallsgrößen verwendet, mit denen Fertigstellungs- und Ausfallzeiten beschrieben werden.
Beispiel:	PearsonType6 alpha1: 2 alpha2: 4 beta: 1

10.2.17 Empirische Verteilungen

Häufig ist die Verteilung von statistischen Eingangsgrößen nicht bekannt. In diesen Fällen gibt es in PACE zwei Möglichkeiten:

1. Anhand von Meßdaten wird versucht, eine mathematische Verteilung finden, die das gemessene oder wenigstens ein ähnliches Verhalten aufzeígt.

2. Aus den Messdaten wird eine empirische Verteilungsfunktion erzeugt.

Die zweite Methode wird verwendet, wenn keine mathematische Verteilung gefunden wird, welche die gemessene Verteilung hinreichend gut approximiert.

10.2.17.1 Erzeugen einer emprischen Verteilung

Für die Erzeugung einer empirischen Verteilung wird das in Abschnitt 6.13.1.4 beschriebene 'Distribution Histogram' verwendet, in das die Meßwerte eingelesen werden.

Bei der Bestimmung einer empirischen Verteilung wird zunächst ein Diagrammfenster für ein 'Distribution Histogram' unter Verwendung des 'view'-Menüs in der PACE-Hauptleiste erzeugt und mit den Menüfunktionen des Fensters geeignet konfiguriert (skalieren, Balkenzahl ausgewählen, usw.).

Danach werden mit der Menüfunktion 'read data file' die Messwerte eingelesen. Diese werden von einer Zeichendatei eingelesen und sind in dieser Datei durch Leerzeichen (blanks), Kommata oder Zeichenvorschübe voneinander zu trennen.

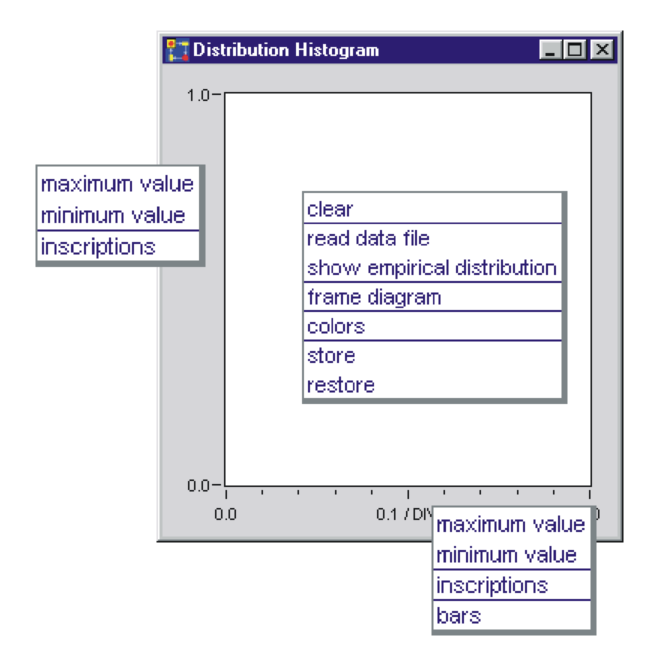


Abb. 10-15: Distribution Histogram mit Menüs

Für die Erzeugung einer möglichst "glatten" Dichtefunktion ist die Anzahl der Balken (bar-Funktion) wichtig. Diese Anzahl sollte so gewählt werden, daß eine geschlossene Balkendarstellung entsteht und nebeneinander stehende Balken möglichst nur kleine Höhenunterschiede aufweisen. Wählt man die bar-Funktion an, so öffnet sich das in Abb. 10-15 dargestellte Fenster, in dem die gewünschte Anzahl von Balken anzugeben ist.

Ist man nach dem Einlesen der Datei mit der Darstellung nicht zufrieden, so ist die Anzahl der Balken zu verändern und die Datei ist erneut einzulesen. Beispielsweise wird man im allgemeinen bei starken Höhenunterschieden benachbarter Balken die Anzahl der Balken verkleinern.

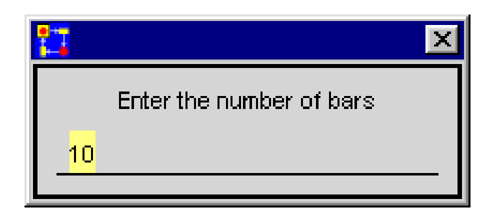


Abb. 10-16: Eingabe der Balkenzahl

Einen Überblick über das aktuelle Aussehen der empirischen Verteilung kann man sich fallweise über die Menüfunktion 'show empirical distribution' verschaffen. Diese zeigt die empirische Dichtefunktion und die empirische Verteilungsfunktion an.

Ist man mit der Gestalt der Dichtefunktion zufrieden, so wird das Histogramm mit der store-Menüfunktion abgespeichert (siehe Abschnitt 6.13.5).

Zu einem späteren Zeitpunkt kann man die empirische Verteilung entweder durch Laden mit der 'restore'-Menüfunktion eines 'Distribution Histogram's oder über Menüfunktionen im 'Evaluator'-Menü der PACE-Hauptleiste wieder anzeigen.

10.2.17.2 Verwenden einer empirischen Verteilung

Zur Erzeugung von Zufallszahlen mit einer empirischen Verteilung wird zunächst mit der Methode:

EmpiricalDistribution name: name-der-verteilung

eine Instanz der empirischen Verteilung erzeugt. name-der-verteilung ist der Dateiname der zugeordneten Datei im Verzeichnis 'ioutils' ohne Erweiterung in Form eines String.

Mit der Methode:

EmpiricalDistribution name: name-der-verteilung
seed: seedNumber

kann eine reproduzierbare Folge von Zufallszahlen erzeugt werden. Ist seedNumber > 0, so erzeugt die Methode 'next' für jede ganze Zahl seedNumber eine jeweils reproduzierbare Sequenz von Zufallszahlen. Für seedNumber = 0 sind die Sequenzen immer voneinander verschieden. Die Methode ist in diesem Fall identisch mit der Erzeugungsmethode ohne die seedNumber-Angabe.

Danach können mit der Methode **next** Zufallszahlen (variates) erzeugt werden.

Falls der Anwender eigene Ströme von Zufallszahlen verwenden will, so kann er eine der Methoden:

EmpiricalDistribution name: name-der-verteilung
use: aRandom

oder

EmpiricalDistribution name: name-der-verteilung
use: aRandom
seed: seedNumber

einsetzen. aRandom wird mit der Klasse Random (siehe PACE-Smalltalk-Fibel) erzeugt.

10.3 Beispiele stochastischer Petri-Netze

Die folgenden Beispiele zeigen die Verwendung von Wahrscheinlichkeitsverteilungen im Zusammenspiel mit Elementen von Petri-Netzen.

10.3.1 Zeichnen von Verteilungsfunktionen

Eine Wahrscheinlichkeitsverteilungen läßt sich leicht (ohne Petri-Netze) unter Verwendung der PACE-Kurven mit der jeweiligen density-Methode zeichnen. Um das Zusammenspiel zwischen Petri-Netzen und Wahrscheinlichkeitsverteilungen an einem einfachen Beispiel zu demonstrieren, sollen im folgenden Liniendiagramme dafür verwendet werden.

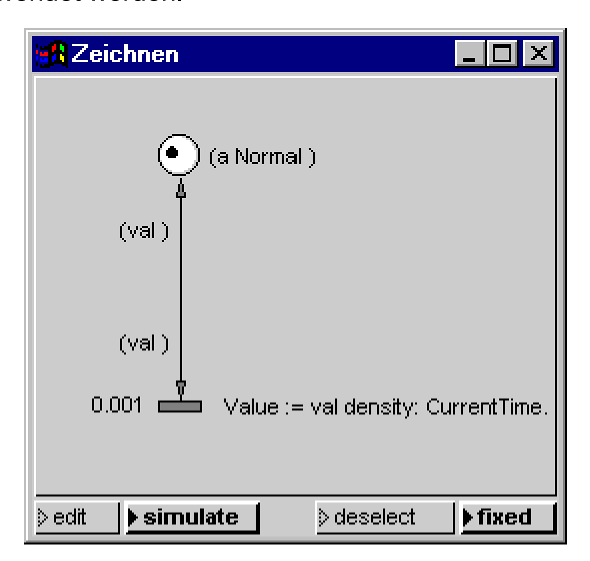


Abb. 10-17: Zeichnen einer Wahrscheinlichkeitsverteilung

Man kann dazu z.B. das in Abb. 10-17 dargestellte Netz verwenden.

In die Stelle wird eine Initialmarke eingelegt, die mit der Wahrscheinlichkeitsverteilung, die gezeichnet werden soll, initialisiert wird. Beim

Ablauf des Netzes wird die Verteilung über das Markenattribut ‘val’ zur Transition transportiert, wobei zwischen zwei Transporten eine Zeitspanne verstreichen soll, die der Auflösung, mit der die Verteilung gezeichnet werden soll, entspricht. Die globale Variable ‘CurrentTime’ gibt damit die Werte der Abszisse vor.

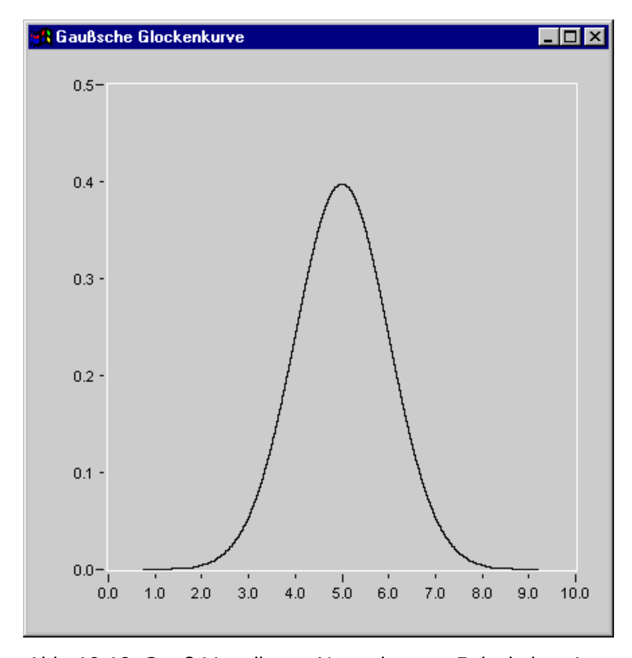


Abb. 10-18: Gauß-Verteilung: Normal mean: 5 deviation: 1

Die zugehörigen Ordinaten wird unter Verwendung der Funktion ‘density:’ berechnet und weist den Wert einer globalen Variablen ‘Value’ zu.

Danach wird an die Stelle ein Liniendiagramm angeschlossen und die ‘resolution’ so eingestellt, daß sie zu der durch die Wartezeit angegebenen Auflösung paßt. Die ‘function’ des Liniendiagramms wird wie folgt geändert:

[:marking :interval | CurrentTime - interval @ Value]

Wenn das Netz abläuft, entsteht die in Abb. 10-18 dargestellte Zeichnung der Gaußschen-Glockenkurve.

10.3.2 Warteschlangen

Für die Modellierung von parallel ablaufenden Prozessen, die auf dieselben Resourcen zugreifen, sind Warteschlangen von größter Bedeutung. Hier wird eine einfache Warteschlange mit einem Bediener behandelt. Weitere Warteschlangen findet man in den Modellen im Verzeichnis ‘samples’ auf der PACE-CD.

Ausgangspunkt ist ein einfacher Betriebsablauf, z.B. die Patientenbedienung bei einem Arzt, der, grob gesehen, wie folgt beschrieben werden kann: durch eine Eingangstüre kommen die Patienten und warten, wenn schon Patienten vorhanden sind, im Wartezimmer auf ihre Behandlung. Die Behandlung wird eingeleitet durch den Aufruf ‘Der Nächste bitte!’, woraufhin der Patient, der von den Wartenden zuerst gekommen war, in das Behandlungszimmer geht. Nach der Behandlung verlässt der Patient die Praxis durch die Ausgangstür.

Zur Darstellung des beschriebenen Vorgangs sind folgende Einzelschritte (Transitionen) notwendig:

1. Ankunft durch die Eingangstür
2. “Der nächste bitte!”
3. Behandlung
4. Verlassen der Praxis durch die Ausgangstür.

Die Wartepositionen (Stellen) fügen sich ganz zwanglos zwischen den Einzelschritten ein:

zwischen 1. und 2. Wartezimmer
zwischen 2. und 3. Behandlungszimmer
zwischen 3. und 4. Ausgang.

In Abb. 10-19 ist das zugeordnete PACE-Petri-Netz dargestellt.

Wenn dieses Netz abläuft, so stellen wir fest, daß ständig Patienten eintreffen für deren Behandlung aber keine Zeit benötigt wird.

Das ständige Erzeugen von Patienten resultiert aus der Tatsache, daß eine Transition, die nur Ausgangsstellen besitzt, ständig Marken erzeugt. Es müssen also Aussagen darüber machen, wie die Patienten zeitlich verteilt eintreffen. Aus dem gleichen Grund wird bisher für die Behandlung der Patienten keine Zeit benötigt. Es muss festgelegt werden, wie lange die Behandlung von Patienten dauert.

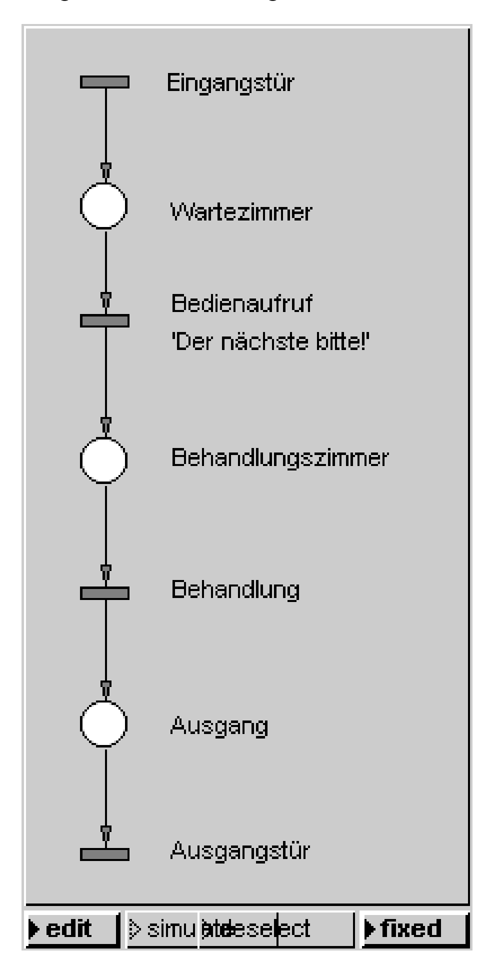


Abb. 10-19: Ausgangsnetz für die Warteschlange

An dieser Stelle lassen sich zwei wichtige Anwendungen von Simulationsmodellen erkennen. Würde man von einer bestimmten Patientenbelegung des Wartezimmers ausgehen und würde man für jede Krankheit eine bestimmte mittlere Behandlungszeit festlegen, so könnte man eine bestimmte Situation simulieren und damit den Ablauf in der Arztpraxis mit allen Daten (z.B. Wartezeiten der einzelnen Patienten) näherungsweise bestimmen.

Obwohl dieser Fall hier relativ trivial anmutet, lässt er sich doch auch leicht direkt ausrechnen, ist er bei komplexeren Aufgabenstellungen wie bei Dispositionssystemen oder bei umfangsreichen Produktionsprozessen sehr nützlich. Man denke beispielsweise an Vergütungsprozesse, in denen sehr viele unterschiedliche Teile verschiedenen Typs von verschiedenen Kunden gleichzeitig bearbeitet werden und bei denen man einem Kunden gegenüber Aussagen über den Fertigstellungstermin seines Auftrags machen soll. Außerdem kann die Auslastung einzelner Teile der Produktionsanlage, können die Kosten pro Vergütungsvorgang und können weitere wichtige Plandaten wie z.B. der voraussichtliche Personalbedarf in verschiedenen Anlagenteilen zu verschiedenen Zeiten schon bei Produktionsbeginn festgestellt werden (Monitoring und Trendsimulation).

Die zweite Anwendungsmöglichkeit, die hier weiter verfolgt werden soll, ist das Ableiten statistischer Aussagen aus der Simulation. Sie wird hauptsächlich bei der Planung und Auslegung eingesetzt. Im vorliegenden Beispiel wird davon ausgegangen, daß die Zeiten zwischen dem Eintreffen von Patienten näherungsweise exponentiell verteilt sind.

Eine Möglichkeit für die Modellierung der Patientenankunft ist in Abb. 10-20 dargestellt. In die Stelle ‘Ankunftszeitintervalle’ wird vor Beginn der Simulation eine Initialmarke gelegt und dieser wird eine Rechenvorschrift, im vorliegenden Fall die Berechnung eines Wertes einer Exponentialverteilung, zugeordnet. Wenn eine Marke von der Stelle ‘Ankunftszeitintervalle’ zu der Transition ‘Eingangstür’ fließen soll, wird die Rechenvorschrift dem Marken-Attribut e zugewiesen und über den Konnektor zur Transition ‘Eingangstür’ transportiert.

Dort wurde als Delay-Code der Smalltalk-Ausdruck ‘e next’ angegeben. Er besagt, daß die zeitliche Verzögerung bis zum Eintreffen des nächsten Patienten durch den nächsten Wert aus der Rechenvorschrift bestimmt sein soll. Wenn diese Zeit abgelaufen ist, wird eine Marke mit der Rechenvorschrift in die Stelle ‘Ankunftszeitintervalle’ zurückgeführt; sie wird zur Berechnung des nächsten Verzögerungsintervalls benötigt. Eine weitere Marke läuft zur Stelle ‘Wartezimmer’ und stellt den nächsten Patienten dar.

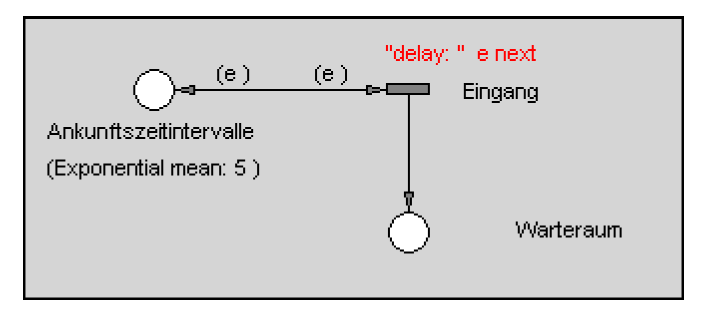


Abb. 10-20: Modellierung zufälliger zeitlicher Verzögerungen

Geht man bei der Modellierung der Behandlungszeiten in der gleichen Weise vor, so gelangt man zu dem in Abb. 10-21 dargestellten Netz.

Durch die Simulation soll herausgefunden werden, wie stark das Wartezimmer belegt ist. Dazu wird an die Stelle ‘Wartezimmer’ ein Histogramm angeschossen. Die Simulation soll im sog. Hintergrundmodus, d.h. mit größtmöglicher Geschwindigkeit, ablaufen.

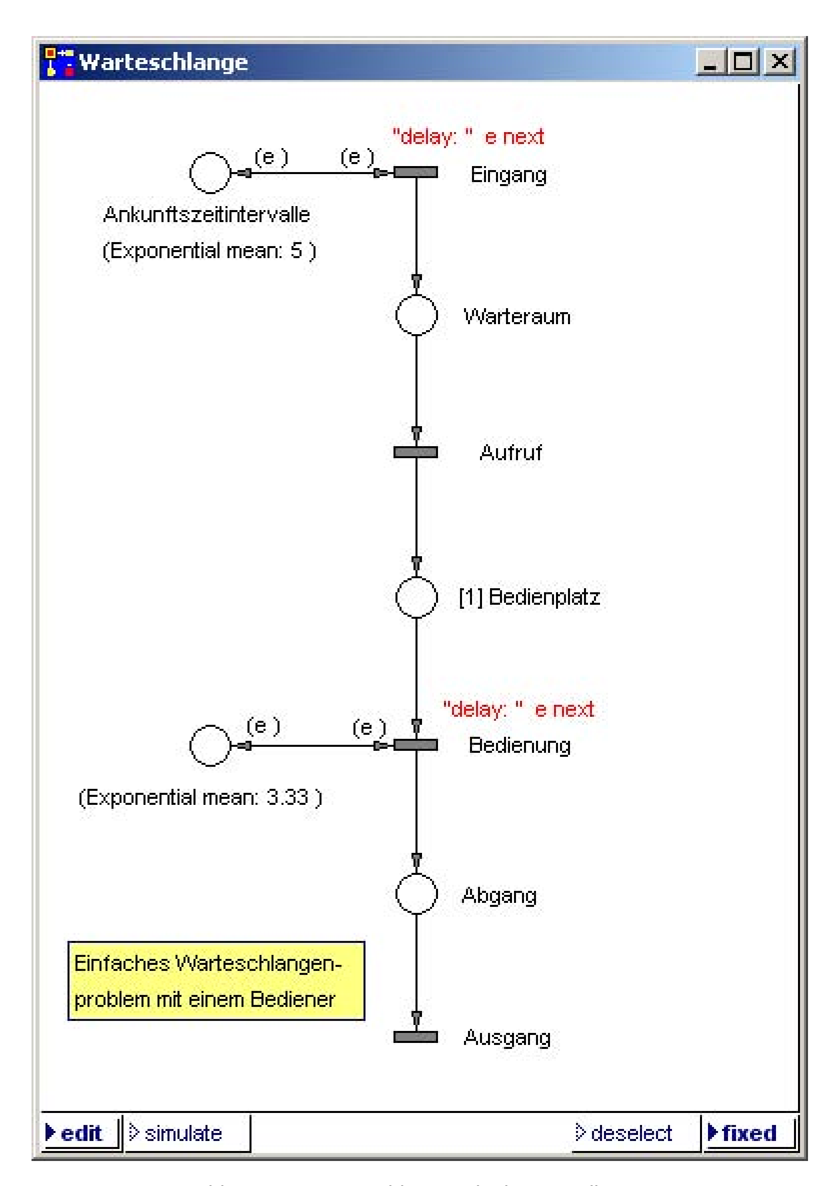


Abb. 10-21: Warteschlange mit einem Bediener

Es entsteht das in Abb. 10-22 dargestellte Value-Histogramm. In ihm ist auf der Abszisse die Anzahl der wartenden Patienten aufgetragen. Die Ordinate zeigt den Anteil der Gesamtzeit von 11000 Zeiteinheiten an, in denen sich die Anzahl von Patienten im Wartezimmer befunden haben.

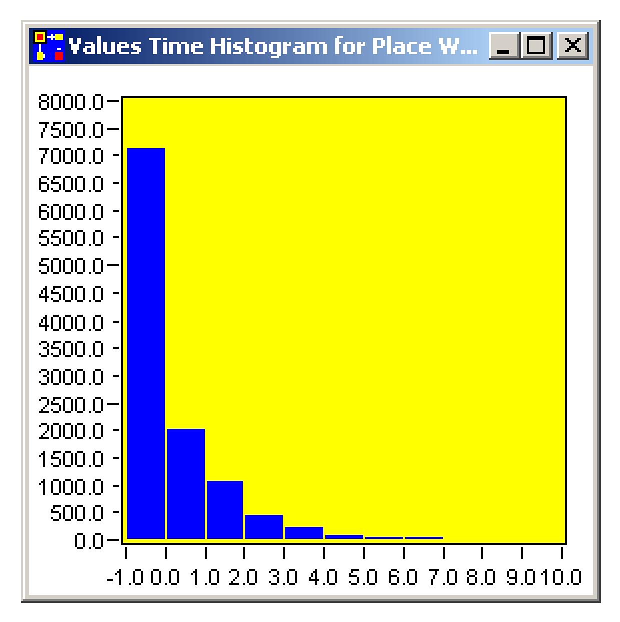


Abb. 10-22:Histogramm über die Belegung des Wartezimmers

10.3.3 Erzeugen einer empirischen Verteilung

Für die Erzeugung einer empirischen Verteilung werden Messdaten benötigt. Da hier nur das Verfahren demonstriert werden soll, wird im folgenden aus den früher beschriebenen mathematischen Verteilungen eine Datei mit Zufallszahlen erzeugt, die dann in ein 'Distribution Histogram' eingelesen wird.

Dazu wird der folgende Code in einem Workspace ausgeführt:

```
"Erzeugen einer Datei mit Randomzahlen"
| ran dataFile stream aString |
```

```
dataFile := 'test.txt' asFilename.
stream := dataFile writeStream.
ran := Normal mean: 3 deviation: 1.
1 to: 250 do: [:i| aString := ran next printString.
                    stream nextPutAll: aString.
                    stream nextPut: $ ].
ran := Exponential mean: 1.
1 to: 250 do: [:i| aString := ran next printString.
                    stream nextPutAll: aString.
                    stream nextPut: $ ].
stream close
```

Die „Messdaten“ bestehen aus einer Mischung der Zufallswerte aus einer Gaußverteilung und einer Exponentialverteilung. Man erhält eine Datei test.txt mit 500 Zufallszahlen, welche die einzulesenden „Messdaten“ darstellen.

Als nächstes wird ein 'Distribution Histogram' erzeugt und wie in Abb. 10-23 skaliert.

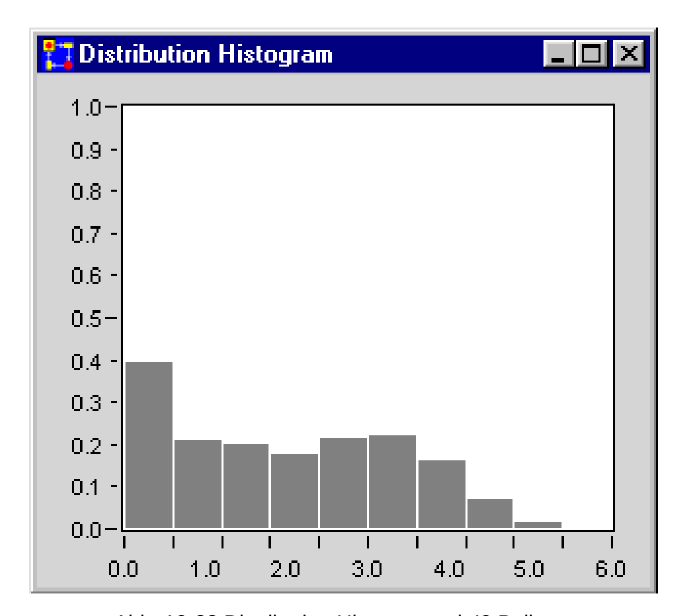


Abb. 10-23:Distribution Histogram mit !0 Balken

Verändert man die Anzahl der Balken im Diagramm 10-23 und liest jeweils die oben erzeugte Datei 'test.txt' ein, dann erhält man für verschiedene Balkenzahlen die in Abb. 10-24 angegebenen, jeweils mit 'show empirical distribution' erzeugten Bilder.

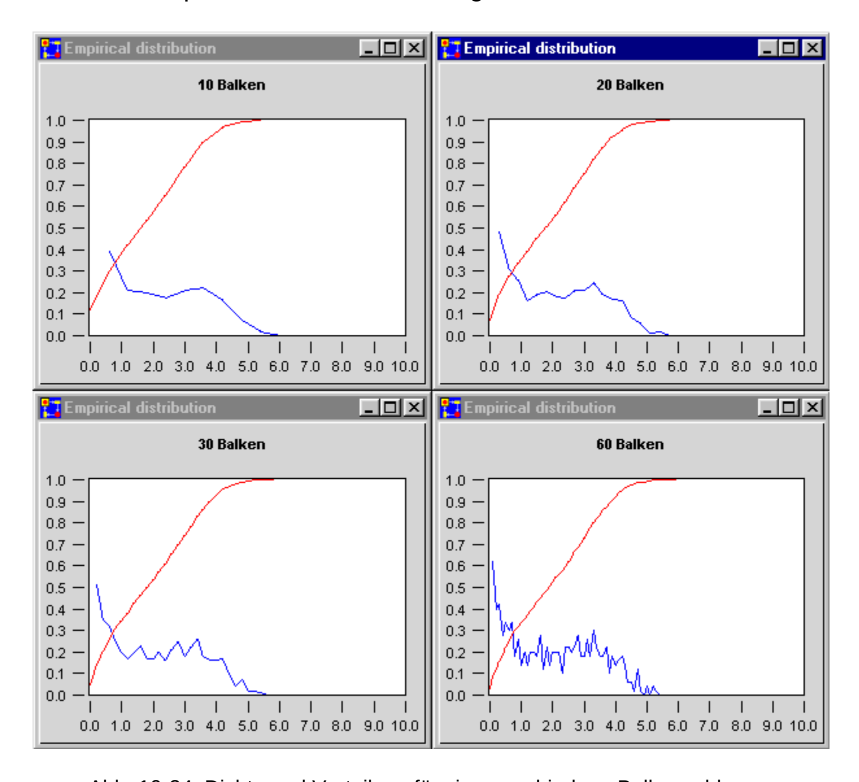


Abb. 10-24: Dichte und Verteilung für vier verschiedene Balkenzahlen

Zufallszahlen werden unter Verwendung der Umkehrfunktion der rot gezeichneten Verteilungsfunktion berechnet. Diese ändert sich nur wenig, wenn man die Anzahl der Balken verändert.

Gewählt wird die Verteilung mit 10 Balken. Das 'Distribution Histogram' wird mit der 'store'-Funktion unter dem Namen 'NeueVerteilung' abgespeichert.

Als nächstes soll verifiziert werden, dass die Verwendung der soeben erzeugten empirischen Verteilung Zufallszahlen (variates) liefert, die der Verteilung der eingelesenen „Messwerte“ entspricht. Für diese Konsistenzprüfung werden in ein 'Distribution Histogram' aus der empirischen Verteilung erzeugte Zufallszahlen eingelesen.

Im 'view'-Menü der PACE-Hauptleiste wird erneut ein 'Distribution Histogram' erzeugt und wie in Abb. 10-23 konfiguriert. Dann wird das folgende Codestück in einem Workspace ausgeführt:

```
"Erzeugen eines DistributionHistograms
 aus der erzeugten empirischen Distribution."
|histo verteil|
verteil := EmpiricalDistribution name: 'NeueVerteilung'.
histo := DistributionHistogram named: 'Distribution Histogram'.
1 to: 1000 do: [:i| histo addValue: verteil next].
```

Wir erhalten das in Abb. 10-20 dargestellte Ergebnis.

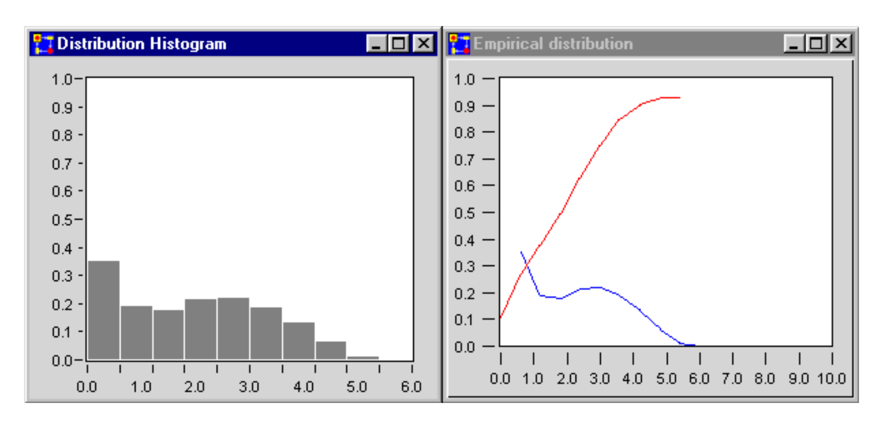


Abb. 10-25: Erzeugen aus einer empirischen Verteilung

Ersichtlich kann man durch Erhöhen der Anzahl von Meßwerten (in vorliegenden Fall der Schleifen-Durchlaufzahl in dem Codestück, das die verwendeten "Messwerte" erzeugt hat) zu genaueren Ergebnissen gelangen. Das hier gezeigte Beispiel zeigt aber, dass man

schon mit relativ wenig Messwerten akzeptable Resultate erzielen kann.

11 STANDARD MODULE

Das folgende Kapitel beschreibt die im Verzeichnis 'modules' des PACE-Verzeichnis gespeicherten Bibliotheksmodule einer PACE-Auslieferung, soweit diese nicht schon an anderer Stelle in diesem Manual beschrieben sind. Die vorgesehenen Module sollen die Modellierung häufig vorkommender Aufgabenstellungen erleichern. Ein Beispiel, das die Vorgehensweise bei Verwendung eines Moduls illustriert, findet sich im PACE Kochbuch, Abschnitt 7.1.

11.1 Priority.sub

Prioritäten werden in PACE über die Attributierung von Marken realisiert. Durch ein zusätzliches Attribut kann die Priorität einer Marke festgelegt werden. Üblicherweise wird die Priorität durch eine positive Ganzzahl n >= 0 festgelegt, wobei die Priorität mit zunehmenden Zahlenwerten abnimmt.

Nach Einsetzen des Moduls 'Priority' (Datei: Priority.sub) aus dem Verzeichnis 'modules' liegt das in Abb. 11-1 dargestellte Teilnetz vor.

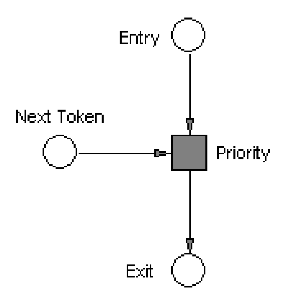


Abb. 11-1: Teilnetz nach Einsetzen des Moduls ''Priority'

Es enthält den in Abb. 11-2 dargestellten Modul 'Priority'.

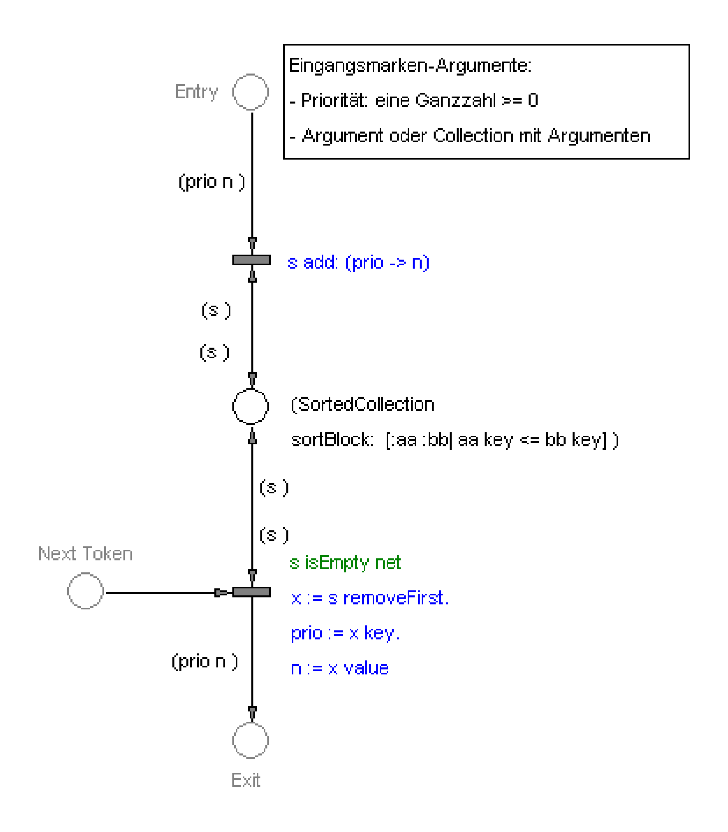


Abb. 11-2: Der Modul "Priority'

Über die Stelle 'Entry' einlaufende Marken haben zwei Attribute, nämlich:

- die Priorität 'prio'
- und ein Datenattribut, welches ein einzelnes Argument oder eine Collection mit Argumenten enthält.

Falls mehrere einzelne Attribute außer dem Prioritätsattribut einer einlaufenden Marke angegeben sind, so muß der Modul geringfügig

erweitert werden (Zusammenfassen der Attribute in einer weiteren Collection).

Die Attribute einlaufender Marken werden nach Prioritäten sortiert in der SortedCollection gespeichert, die in der mittleren Stelle in Abb. 11-2 als Attribut einer Initialmarke erzeugt wird. Durch Einlegen einer Marke in die Stelle 'Next Token' kann der Anwender eine Marke mit den zur höchsten Priorität gehörigen Attributen erzeugen, welche in die Stelle 'Exit' läuft, von wo sie vom Anwendernetz verbraucht werden kann.

11.2 LIFO.sub

Normalerweise verlassen die Marken eine Stelle in der Reihenfolge, in der sie eingetroffen sind (FIFO = First In First Out). Gelegentlich besteht aber der Wunsch, die Marken in umgekehrter Reihenfolge zu bearbeiten (LIFO = Last In First Out). Das kann mit dem Modul LIFO.sub bewerkstelligt werden.

Nach Einsetzen des Moduls 'LIFO' (Datei: LIFO.sub) aus dem Verzeichnis 'StandardModules' liegt das in Abb. 11-3 dargestellte Teilnetz vor.

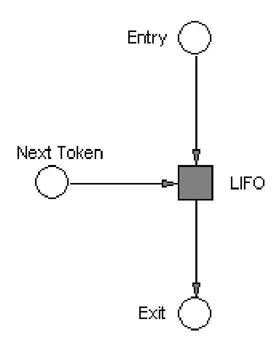


Abb. 11-3: Teilnetz nach Einsetzen des Moduls ‘LIFO'

Der darin enthaltene Modul ‘LIFO' ist in Abb. 11-4 dargestellt.

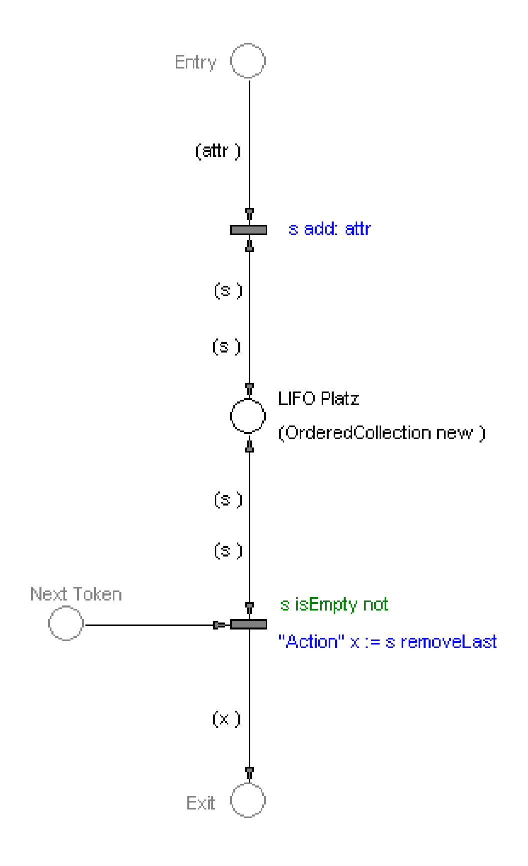


Abb. 11-4:Der Modul ‘LIFO'

Dabei wird angenommen, daß als Markenattribute ‘attr’ entweder ein einzelnes Argument oder eine die Daten enthaltene Collection vorliegt. Liegen mehrere einzelne Argumente vor, so ist der Modul geringfügig zu erweitern.

Die Attribute einlaufender Marken werden der Reihe nach in der OrderedCollection gespeichert, die in der mittleren Stelle in Abb. 11-4 als Attribut einer Initialmarke erzeugt wird. Durch Einlegen einer

Marke in die Stelle 'Next Token' kann der Anwender eine Marke mit den Attributen der zuletzt angekommenen Marke erzeugen, welche in die Stelle 'Exit' läuft, von wo sie vom Anwendernetz verbraucht werden kann.

11.3 RandomToken.sub

Mit dem Modul 'RandomToken' (Datei: RandomToken.sub) können Token in zufälliger Reihenfolge bearbeitet werden. Ähnlich wie in den vorangegangenen Abschnitten erhält man nach dem Einsetzen des Moduls zunächst das in Abb. 11-5 dargestellteTeilnetz.

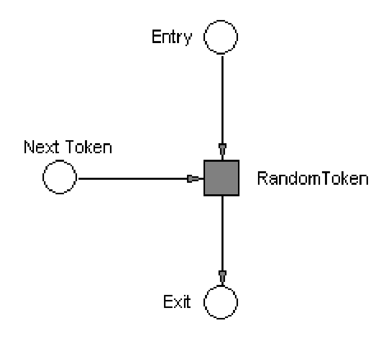


Abb. 11-5: Teilnetz nach Einsetzend des Moduls 'RandomToken'

Der Submodul 'RandomToken' ist in Abb. 11-6 dargestellt. Wie bei den früher beschriebenen Moduln erhält man ein Token mit aus der OrderedCollection zufällig ausgewähltem Attribut durch Einlegen einer Marke in die Stelle 'Next Token'.

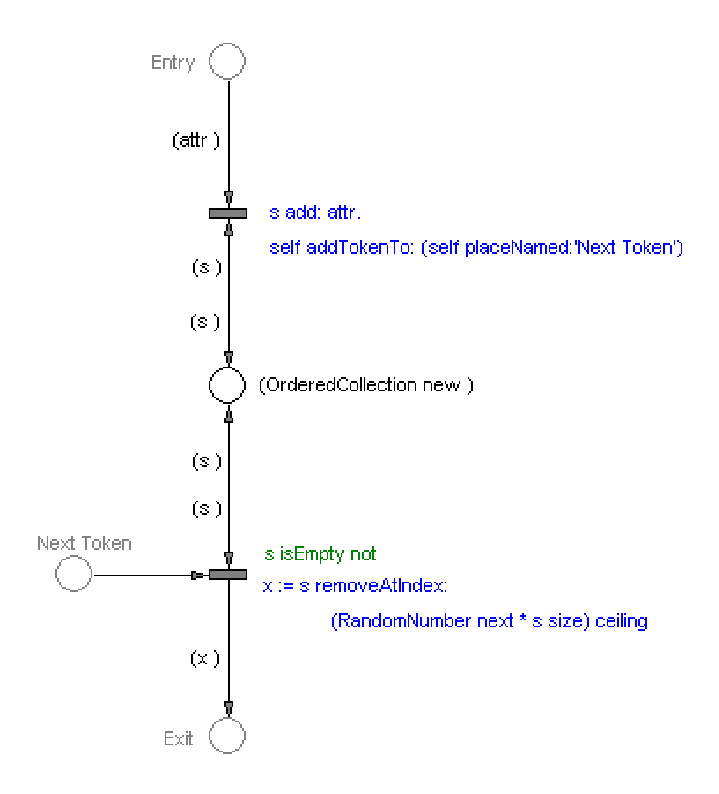


Abb. 11-6: Der Modul 'RandomToken'

12 LITERATUR ÜBER PETRI-NETZE

Dirk Abel: "Petri-Netze für Ingenieure, Modellbildung und Analyse diskret gesteuerter Systeme",
Springer Verlag, Berlin Heidelberg New York, ISBN 3-540-51814-2, 1990.

Jens v. Aspern: "SPS-Softwareentwicklung mit Petrinetzen", Speicherprogrammierbare Steuerungen: Bd. 8, Hüttig
Buch Verlag, Heidelberg, ISBN 3-7785-2197-7, 1993.

Jens v. Aspern: "SPS-Softwareentwicklung Petrinetze und Wortverarbeitung",
Speicherprogrammierbare Steuerungen: Bd. 10, Hüttig Buch Verlag, Heidelberg, ISBN 3-7785-2279-5, 1994.

Bernd Baumgarten: "Petri-Netze: Grundlagen und Anwendungen", BI-Wiss-Verlag, Mannheim Wien Zürich, ISBN 3-411-14291-X, 1990.

J. Billington, M. Diaz, G. Rozenberg (Eds.): "Application of Petri-Nets to Communication Networks"", Springer Verlag Berlin Heidelberg NewYork (1999), ISBN 3-540-65870-X

H. Eggert "Die Automatisierung eines
technischen Prozesses zur Bierherstellung mit einer Prozeß-Datenverarbeitungsanlage", PDV-Berichte KfK-PDV 166, Kernforschungszentrum Karlsruhe, Dezember 1978.

Hans-Michael Hanisch: "Petri-Netze in der Verfahrenstechnik: Modellierung und Steuerung verfahrenstechnischer Systeme", Oldenburg Verlag, München, ISBN 3-486-22190-6, 1992.

Wolfgang Hümbs, Klaus Kuzyk: "Modellierung paralleler Prozesse durch Petri-Netze", mc Microcomputer-Zeitschrift, S. 54-58, August 1989.

IEEE Transactions on Semiconductor Manufacturing: "Special section on Petri Nets in Semiconductor Manufacturing", Volume 11, Number 3, August 1998, ISSN 0894-6507

Kurt Jensen: "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", Volume 1, Springer Verlag, Berlin Heidelberg New York, ISBN 3-540-55597-8, 1992.

Kurt Jensen: "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", Volume 2, Springer Verlag, Berlin Heidelberg New York, ISBN 3-540-58276-2, 1995.

Kurt Jensen: "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", Volume 3, Springer Verlag, Berlin Heidelberg New York, ISBN 3-540-62867-3, 1997.

Wolfgang Jung: "Parallelsteuerung und Petri-Netze", mc Microcomputer-Zeitschrift, S. 106-113, Juni 1990.

Anastasia Pagnoni: "Project Engineering, Computer-Oriented Planning and Operational Decision Making", Springer Verlag, Berlin Heidelberg New York, ISBN 3-540-52475-4, 1990.

J. L. Peterson: "Petri Net Theory and the Modelling of Systems", Prentice Hall Inc., Engelwood Cliffs, NJ 07632, 1981.

Carl Adam Petri: "Kommunikation mit Automaten", Dissertation, erschienen in: Schriften des Rheinisch-Westfälischen Instituts für instrumentelle Mathematik an der Universität Bonn, Bonn 1962.

Wolfgang Reisig: "Systementwurf mit Netzen", Springer Verlag, Berlin, Heidelberg, 1985.

Wolfgang Reisig: "Petrinetze - Eine Einführung",
Springer Verlag, Berlin, Heidelberg New York, 2. Auflage, ISBN
3-540-16622--X, 1991.

Wolfgang Reisig, Grzegorz Rozenberg (Eds.): "Lectures on Petri
Nets I: Basic Models", Springer Verlag Berlin Heidelberg NewYork
(1998), ISBN 3-540-65306-6

Wolfgang Reisig, Grzegorz Rozenberg (Eds.): "Lectures on Petri
Nets II: Applications", Springer Verlag Berlin Heidelberg NewYork
(1998), ISBN 3-540-65307-4

B. Rosenstengel, U. Wienand: "Petri-Netze - Eine anwendungsorien-
tierte Einführung",
Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden, 4. Auflage, ISBN
3-528-33528-3, 1991.

Eckehard Schnieder (Hrsg.): "Petri-Netze in der Automatisierungs-
technik", Oldenburg Verlag, München Wien, ISBN
3-486-22045-4, 1992.

Peter H. Starke: "Petrinetze", Deutscher Verlag der Wissenschaften,
Berlin, 1980.

Peter H. Starke: "Analyse von Petri-Netz-Modellen",
Teubner Verlag Stuttgart, ISBN 3-519-02244-3, 1990.

Konrad Zuse: "Petri-Netze aus der Sicht des Ingenieurs",
Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden, 1980.

Konrad Zuse: "Anwendungen von Petri-Netzen",
Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden, 1982.

Stichwortverzeichnis

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z